AD-A150 750   SEMANTIC NETWORKS AND NEURAL NETS(U) ROCHESTER UNIV NY   1/2
DEPT OF COMPUTER SCIENCE L SHASTRI ET AL. JUN 84
TR-131 N00014-82-K-0193

UNCLASSIFIED   F/G 9/4   NL
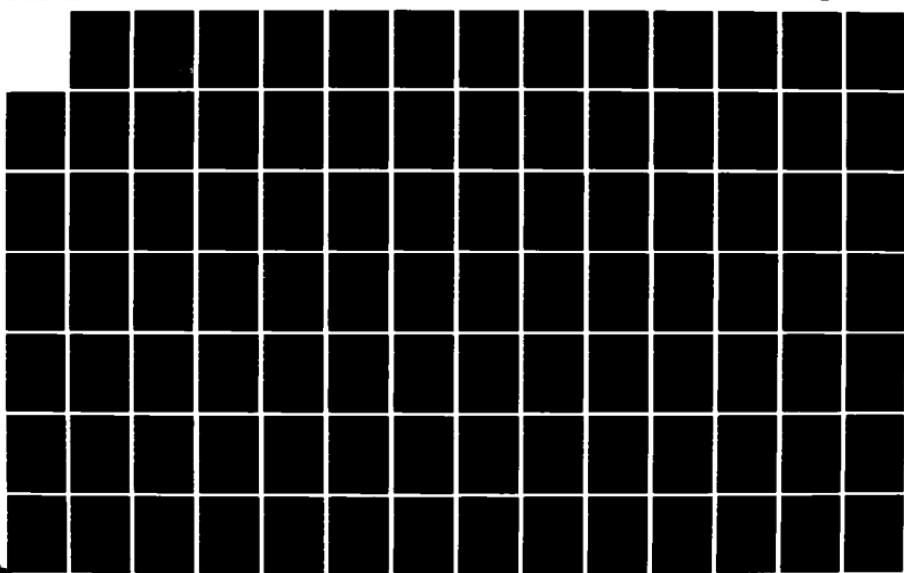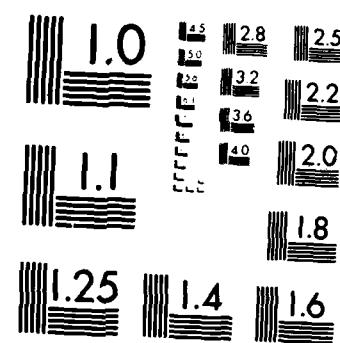
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

③

AD-A150 750

rochester

Department of Computer Science
University of Rochester
Rochester, New York 14627

85 02 11 136

Semantic Networks and Neural Nets

Lokendra Shastri and Jerome A. Feldman
Computer Science Department
The University of Rochester
Rochester, NY 14627

TR131
June, 1984

Abstract

_ Connected networks of nodes representing conceptual knowledge are widely employed in artificial intelligence and cognitive science. This report describes a direct way of realizing these semantic networks with neuron-like computing units. The proposed framework appears to offer several advantages over previous work. It obviates the need for a centralized knowledge base interpreter, thereby partially solving the problem of computational effectiveness and also embodies an evidential semantics for knowledge that provides a natural treatment of defaults, exceptions and "inconsistent" or conflicting information. The model employs a class of inference that may be characterized as working with a set of competing hypotheses, gathering evidence for each hypothesis and selecting the best among these. The resulting system has been simulated and is capable of supporting existing semantic network applications dealing with problems of recognition and recall in a uniform manner.

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER TR 131	2. GOVT ACCESSION NO. AD-A150 750	3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle) Semantic Networks and Neural Nets	5. TYPE OF REPORT & PERIOD COVERED technical report
	6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s) Lokendra Shastri and Jerome A. Feldman	8. CONTRACT OR GRANT NUMBER(s) N00014-82-K-0193

9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department University of Rochester Rochester, New York 14627	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209	12. REPORT DATE June, 1984
	13. NUMBER OF PAGES 100

14. MONITORING AGENCY NAME & ADDRESS(If different from Controlling Office) Office of Naval Research Information Systems Arlington, Virginia 22217	15. SECURITY CLASS. (of this report) unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distributed of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

semantic networks, evidential semantics, massively parallel computation, neural nets, network models.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Connected networks of nodes representing conceptual knowledge are widely employed in artificial intelligence and cognitive science. This report describes a direct way of realizing these semantic networks with neuron-like computing units. The proposed framework appears to offer several advantages over previous work. It obviates the need for a centralized knowledge base interpreter, thereby partially solving the problem of computational effectiveness and also embodies an evidential semantics for knowledge that provides a natural treatment of defaults, exceptions and "inconsistent" or conflicting

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE
1 JAN 73

information. The model employs a class of inference that may be characterized as working with a set of competing hypotheses, gathering evidence for each hypothesis and selecting the best among these. The resulting system has been simulated and is capable of supporting existing semantic network applications dealing with problems of recognition and recall in a uniform manner.

Accession For

NTIS GRA&I ☒

DTIC TAB ☐

Unannounced ☐

Justification

By

Distribution/

Availability Codes

Avail and/or

Dist   Special

A-1

DTIC COPY INSPECTED 4

1. Introduction

The past few years have witnessed a significant reawakening of interest in massively parallel computation, often portrayed as neural nets. Rapid advances in computational, behavioral and biological theories have brought about a new and much more sophisticated effort to model cognition and perception in physiologically plausible terms. But essentially all of the detailed work has been concerned with relatively peripheral (low conceptual level) activities such as early vision, word recognition, speech and motor control. Higher mental functions such as language comprehension, logical inference and planning have not been treated effectively with massively parallel techniques and many scientists believe that it is impossible to do so. One purpose of this paper is to suggest a particular set of (connectionist) mechanisms for a general representation of conceptual information and for using this information in inferences. These mechanisms appear to form an adequate base for the study of problems of higher-level vision and language understanding.

The representation of complex knowledge and associative access to it are recognized to lie at the core of intelligence. Semantic Networks – graph structures with 'concepts' as nodes and 'associations' as arcs – have become a standard way of envisioning knowledge representation schemes. This paper suggests a unified approach to semantic network representations, which appears to have a number of advantages over previous schemes.

Semantic network models of various kinds have been used in a wide range of studies in artificial intelligence and cognitive science. One line of work uses 'spreading activation' in concept networks to model contextual effects in e.g. word perception [McClelland & Rumelhart 81], disambiguation [Quillian 68; Cottrell & Small 83], speech production [Dell 80] and memory retrieval [Anderson 83]. Most other work using semantic network models assumes that the network is passive and is *interpreted* by a control program. Interpreted semantic networks can be further divided into recognition and deduction applications of networks. Recognition of complex visual scenes is almost universally based on network models at the higher conceptual levels [Ballard & Brown 82; Marr & Nishihara 78] and speech recognition work often has this character [Lowerre & Reddy 79]. Deduction models employing semantic networks are generally employed in natural language and related research [Walker 78; Findler 79].

All of these various uses of semantic networks make somewhat different demands on the representation and have evolved to the point where they have distinct computational characteristics. The most important point for us is the presence or absence of a distinct interpreting program that can examine and modify the network. For a variety of reasons to be outlined below, we will consider only systems with no interpreter and attempt to show how such systems can support all the existing applications of semantic networks. The only computational primitives in our models will be the calculation and transmission of activity states. This is the computational characteristic of neural net models -- whence the title of the paper.

In addition to the general virtues of uniformity we have several technical reasons for exploring activation (or value passing [Fahlman 82]) models of semantic networks. One technical problem arises when the information to be captured

contains apparently inconsistent statements. A well known example is given below. The four facts:

> Dick is a Quaker
> Quakers are pacifists
> Republicans are not pacifists
> Dick is a Republican

could well arise in real life without causing difficulty. In standard semantic network models using conventional logic as a basis, the four statements entail a contradiction which, in the worst case, could render the system useless. More sophisticated non-monotonic logical treatments [Reiter 80], suggest that the system must choose a consistent subset of the four assertions, but this entails making arbitrary choices and amounts to ignoring some of the information provided. *There is no denying that Dick could be a Quaker and a Republican at the same time.* There is some evidence th.t Dick is not a pacifist (because he is a Republican and Republicans *tend* to be non-pacifists) at the same time there is some evidence that he is a pacifist (he is a Quaker and Quakers tend to be pacifists). One goal for our semantic network model is that it be able to incorporate evidential statements like those in the example and to draw appropriate inferences from them. The Republican-Quaker example is typical of a large number of evidential conflict situations. For example: knowing that someone likes Mexican food but dislikes chicken does not specify whether he will like Chicken Mole. But if we know the strength of his likes and dislikes, the question might be easily answered.

Another technical issue that we address in the current model is the specification of 'natural kind' terms. There is no way to specify exactly what conditions are necessary and sufficient to have something be deemed an 'elephant' or 'chair' or 'fight' [Smith & Medin 81]. We will tre.: this issue also as one of evidence. The idea is seen most clearly in the realm of visual input, where various features are recognized with varying degrees of confidence. There is an enormous variety of conditions under which one would be willing to assert the presence of a chair. In our treatment, chair legs are evidence for the presence of a chair in the same way that Republicanism is evidence for non-pacifism. The major difficulty with this approach is the absence of an adequate mathematical theory of evidential reasoning: we will say more about this in Section 5.1.

An evidential framework seems to be useful for both characterizing entities and for dealing with conflicting assertions, but one could pursue evidential formulations without abandoning the interpretive version of semantic nets [Lowrance 82]. Abandoning the interpretive model has several distinct benefits which will be discussed at appropriate places in the paper. For now, we will be content with one consideration which has led us (and others) to concentrate on massively parallel, active networks for modelling intelligent behavior.

Psychological and biological results suggest that many cognitive tasks like visual recognition, categorization and associative retrieval do not take more than 100 sequential steps. This follows because typical neuronal firing rates are a few milliseconds and the response time of cognitive agents during numerous experimental tasks is a few hundred milliseconds. This observation imposes a major

constraint on the manner in which conceptual information may be organized and accessed and is one reason for our rejection of an interpreter. Other motivations for the choice of spreading activation models include their good fit with human data on retrieval [Dosher 83], on errors [Dell 80] and variability and the clear mapping to the underlying physiological substrate.

The massively parallel (connectionist) formulation of semantic networks avoids the interpreter bottleneck, but brings in a whole range of new problems in representation, stability, learning, etc. There is a growing literature dealing with these issues. We will concern ourselves in this paper only with representation questions, using a particular computational model outlined in Section 1.2 and presented in detail in [Feldman & Ballard 82]. There is enough experience with the theoretical and experimental properties of this formulation to convince us that the key computational questions can be resolved. The machine examples in this paper were all built using a general purpose simulator [Small et al. 82] which has also been used in a variety of other domains [Addanki 83; Cottrell & Small 83]. By suppressing the computational issues, we are able to focus on the questions of representation and use of knowledge of which there are plenty. The central question addressed in this paper is the efficacy of a self-activating semantic network as the vehicle for conceptual reasoning.

In our formulation, a semantic network (SN) is an information retrieval mechanism with a limited amount of built-in inference (cf. [Allen 83]). The first issues to be addressed are how a query is presented to a SN and how an answer is returned. In order to keep the treatment uniform, we require that the query be presented and the answer be received in connectionist fashion. This is achieved by introducing a simple kind of routines expressed as connectionist networks. A query arises from a point in a routine where information is needed and answers are returned by activating appropriate units in the inquiring routine. Our routines are, like schemes and scripts, not adequate to model the full range of plans and actions [Schank 82] but will suffice for our purposes (cf. Section 5.2).

We will start with a trivial example to introduce the notation and general framework of our treatment. Figure 1.1 shows a fragment of a simple restaurant routine for a person who always orders one of two lunches depending upon how hungry he is. A routine is represented as a sequence of nodes (units) connected so that activation can serve to sequence through the routine. Stepping from one node in a routine to its successor will depend on a completion signal which will not be shown explicitly. We depict action steps as oval-shaped nodes, queries as hexagonal nodes and answers as circular ones. In this routine, ordering a meal gives rise to a query about the person's state of hunger, directed to the semantic network which this paper is attempting to characterize. Answers are supposed to come from units (as yet unspecified) that are connected to the [yes] and [no] units in the routine. The [yes] and [no] nodes are primed by the question unit to be responsive to activation and are connected in a WTA (winner-take-all) (described later in Section 3.2) fashion to force a decision. Whichever answer node dominates will activate its successor and thus trigger the appropriate speech act. We will obviously be dealing with more complex routines, queries and answers, but the basic structure of this example will be maintained throughout the paper. Some of the questions raised and avoided in this formulation of the problem will be discussed in Section 5.

The routine of Figure 1.1 is particularly simple because the query involves no input information. A somewhat more complex routine fragment is required for someone who orders the daily special if it is a meal that he likes and falls back on his regular order if the special doesn't appeal to him (Figure 1.2). In this case, the query to the Semantic Network must include some situational information, namely the special featured that day. Routines will generally include several such parameters that will need to be specified in the execution of the routine. We call these *roles* and they will play a central part in our treatment. A role will be *bound* to a *concept* for the execution of its routine. In Figure 1.2, the [special] role might be bound to [ham]. We assume that this binding is implemented by a dynamic link network (cf. [Feldman, 82a]) so that activating either end of the [special ~ ham] link will activate the other end. Notice that the binding will permit parameterized actions such as ordering ham when ham is the daily special. Roles correspond to typed or sorted free variables in a formulation based on mathematical logic.

World knowledge, such as one's taste for ham, is encoded in the semantic network (SN) which is the main focus of this paper. For this introduction, we will give a crude overview of how the SN and the luncheon routine would interact in choosing what to order. The various concepts in the SN will be represented by individual nodes, which will be depicted by rectangular boxes. Since there is no interpreter, the links between concepts will have to be tightly specified in their spreading of activation. All of this is worked out in Sections 2-4. The oversimplified network given in Figure 1.3 has each kind of food directly linked to the answers for which it is appropriate. In this case, activating the question [special appeals?] activates the role node for [special] which activates its binding [ham]. The [ham] unit in the role network directly activates the [ham] node in the SN which sends activation to the particular [yes] node of this routine (as well as to many other places). Since this routine and the particular [yes] node are enabled, activation continues to the appropriate response. The dynamic link [special ~ ham] will then cause the word "ham" to be spoken when the [say special] action is carried out.

The remainder of the paper is concerned with the details of this process. Figure 1.4 gives a more accurate indication of the knowledge representation mechanism. The network in Figure 1.4 encodes the following information:

HAM and YAM are two types of objects in the domain.
Objects in the example domain are characterized by two properties, *HAS-TASTE* and *HAS-FOOD-KIND*.
HAM is SALTY in taste and is a kind of MEAT.
YAM is SWEET in taste and is a kind of VEGETABLE.

Each arc in the network represents a pair of links, one in either direction. We are using an arc in place of a pair of links to improve the readibility of these diagrams.

The triangular nodes in the network associate objects, properties and property values.

Each node is a computing element and, when in an "active" state, sends out activation to all the nodes connected to it. A node may become active on receiving activation from another node or an external source. Triangular nodes behave slightly

differently in that they become active only on receiving simultaneous activation from two nodes.

The crude description given above is sufficient to demonstrate how simple recognition and retrieval tasks may be handled by such networks. To find an object in the network with a salty taste one would activate the nodes *HAS-TASTE* and SALTY. The triangular node linking *HAS-TASTE* and SALTY will receive coincident activation along two of its links and become active. As a result, it will transmit activation to HAM which will ultimately become active.

Alternately, assume that one is interested in finding out the taste of HAM. This could be done by activating the nodes *HAS-TASTE* and HAM. This will cause the same triangular node to become active and transmit activation to SALTY. Eventually, SALTY will become active completing the retrieval.

The two examples roughly correspond to the manner in which recognition and retrieval take place in these networks. The detailed description of how questions like these can be treated uniformly in connectionist networks is fairly complex and is broken up into three parts. Section 2 describes our formalization of conceptual knowledge without reference to particular computational mechanisms and may be of independent interest to researchers whose connectionism remains inhibited. The central ideas are the evidence formulation, multiple structural hierarchies and articulating treatments of type-token and attribute-value relationships. Section 3 is concerned with inferences in our interpreter-free model. The basic semantic network is shown to be adequate for associative retrieval and basic categorization judgments, along with some generalized inheritance inferences: It is suggested that routines and their associated role networks suffice for a wide range of further inferences. All of this is done in the absence of computational details, which are presented in Section 4. The computational model includes the specification of the roles of connection and activation for the different classes of nodes in the network and of the strength of weights on connections. Section 5 consists of brief discussions of several important issues that are beyond the scope of the current paper. These include technical questions on convergence and evidence theory, the treatment of complex ontological types such as events, and of diffuse conceptual structures, and the basic problem of learning. This paper, then, fulfills one promise made in earlier connectionist tracts [Feldman & Ballard 82; Feldman 82b] and indicates how several additional pieces might fit into the puzzle.

The next subsection briefly describes the connectionist model and may be skipped if the reader is familiar with the model.

1.2 The Connectionist Model

Connectionism provides a plausible model of computations carried out in neuronal networks and, independent of any such consideration, is a powerful model of massively parallel computation. The details of the connectionist model in its current state of development have been laid out in [Feldman & Ballard 82; Feldman 82a]. We present the salient features below.

Connectionist networks are made up of *active* elements that are capable of

performing simple processing. These units have very high fan-ins and fan-outs and communicate with the rest of the network by transmitting a simple value. A unit transmits the same value to all units to which it is connected. The output value is closely related to the unit's potential and is best described as a level of activation. A unit's potential reflects the amount of activation the unit has been receiving from other units. All inputs are weighed and combined in a manner specified by the *site functions* and the *potential function* in order to update a unit's potential. A more technical description follows.

A network consists of a large number of units connected to a large number of other units via links. The units are computational entities defined by:

$\{q\}$	: a small set of states. (fewer than 10)
p	: a continuous value called potential
v	: an output value. approximately 10 discrete values
i	: a vector of inputs i_1, i_2, ...i_n (this is elaborated below)

together with functions that define the values of potential, state and output at time $t+1$, based on the values at time t:

$$p_{t+1} \longleftarrow P(i_t, p_t, q_t)$$
$$q_{t+1} \longleftarrow Q(i_t, p_t, q_t)$$
$$v_{t+1} \longleftarrow V(i_t, p_t, q_t)$$

A unit does not treat all inputs uniformly. Units receive inputs via links (or connections) and each incoming link has an associated *weight*. A weight may have a *negative* value. A unit weighs each input using the weight on the appropriate link. Furthermore, a unit may have more than one "input site" and incoming links are connected to specific sites. Each site has an associated site-function. These functions carry out local computations based on the input values at the site, and it is the result of this computation that is processed by the functions P, Q and V. The notion of sites is useful in defining interesting unit behavior like AND-of-OR or OR-of-AND. Sites will be used extensively in our solutions. The functions P, Q and V are arbitrary but in keeping with the underlying philosophy of these models, it is desired that these functions be "simple." The details of the functions used in this paper are presented in Section 4; only the general form of the computation is needed for the next two sections.

FIGURE 1.1

ROLE

SPECIAL ~ HAM

SEMANTIC NETWORK

ORDER MEAL → SPECIAL APPEALS → YES → ORDER SPECIAL

NO → ORDER REGULAR

ONE ROLE SPECIAL
IS BOUND

ORDER SPECIAL IF APPEALING
ELSE REGULAR
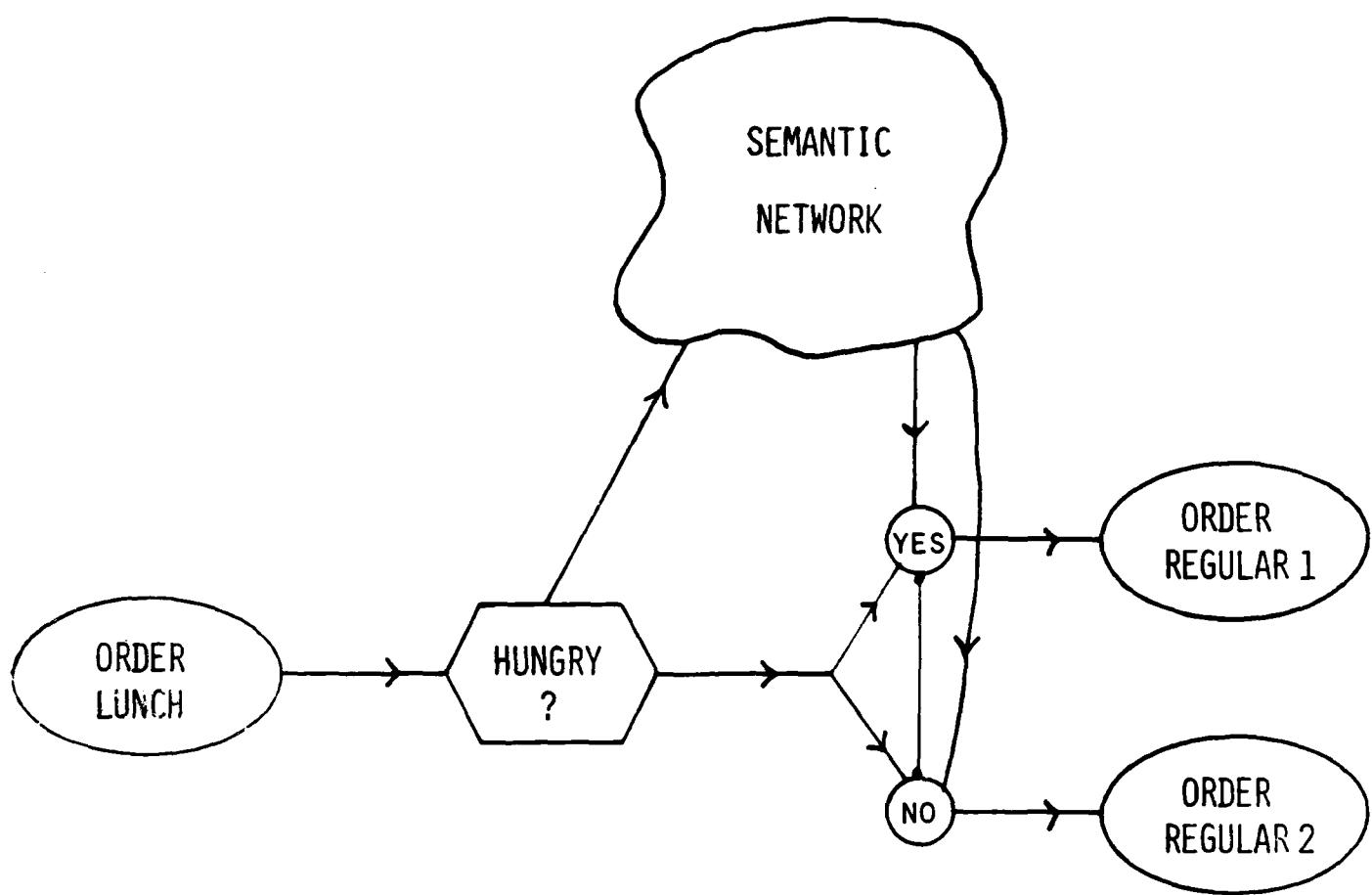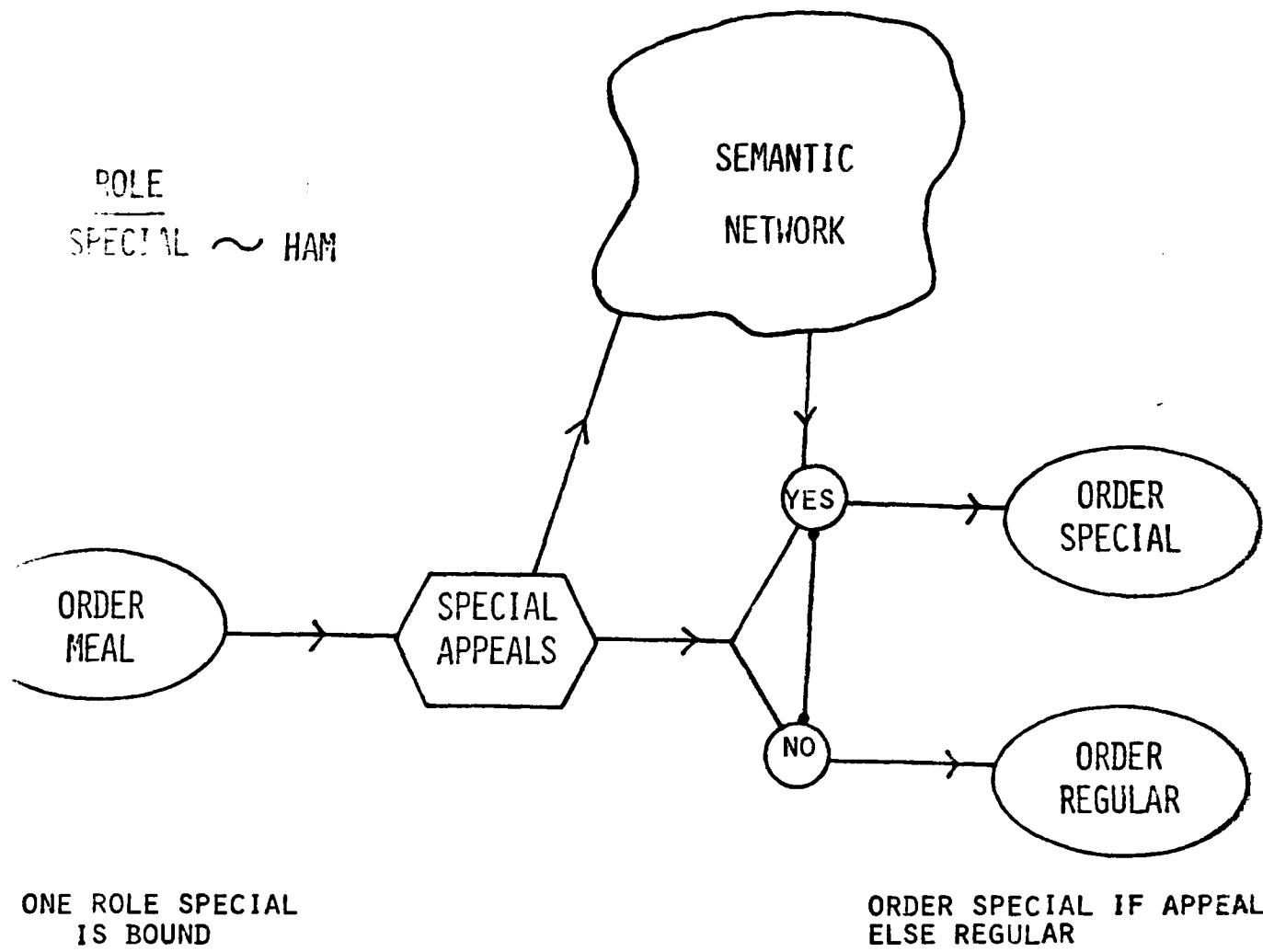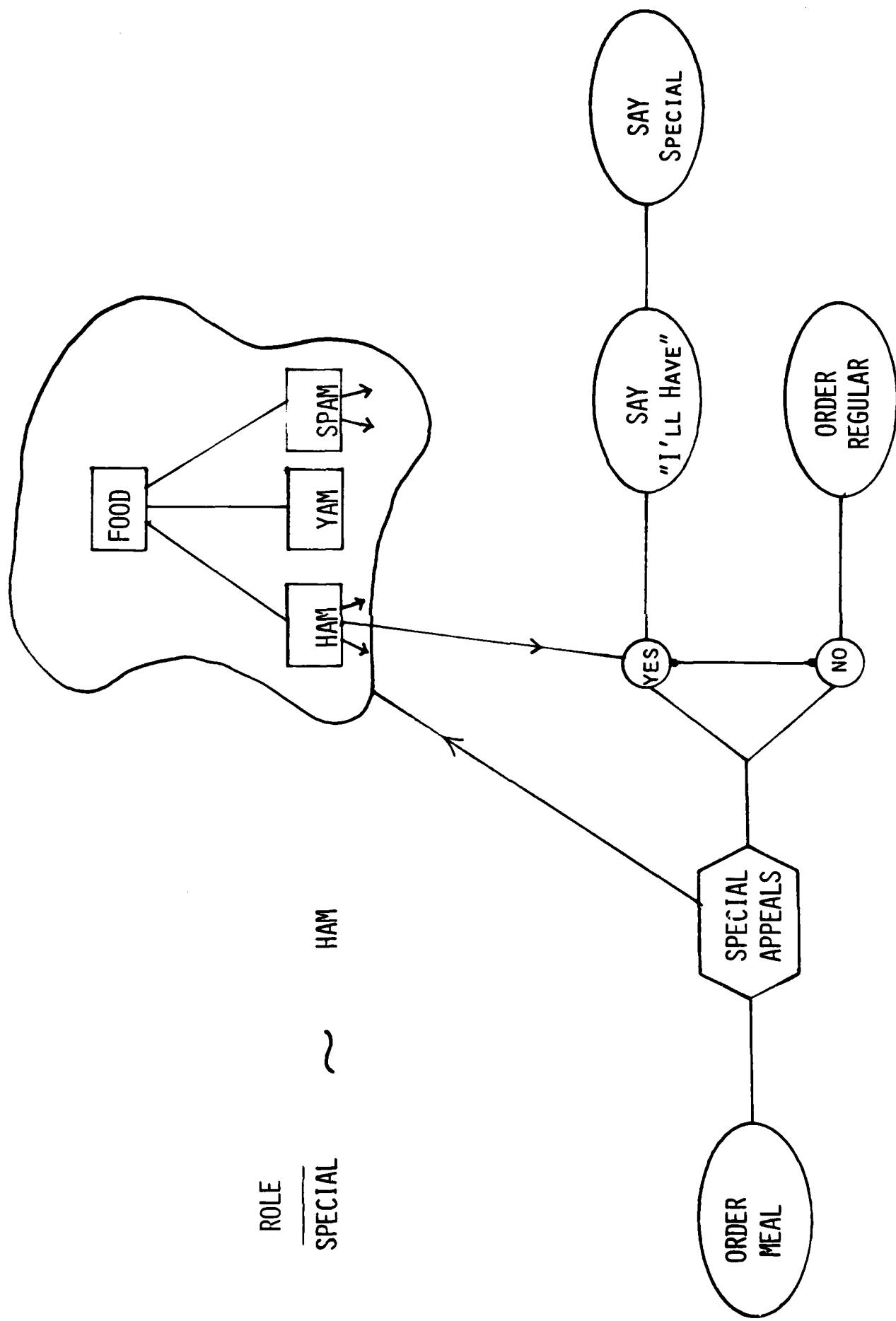
FIGURE 1.2

FURTHER DETAIL ON ORDERING THE SPECIAL

FIGURE 1.3
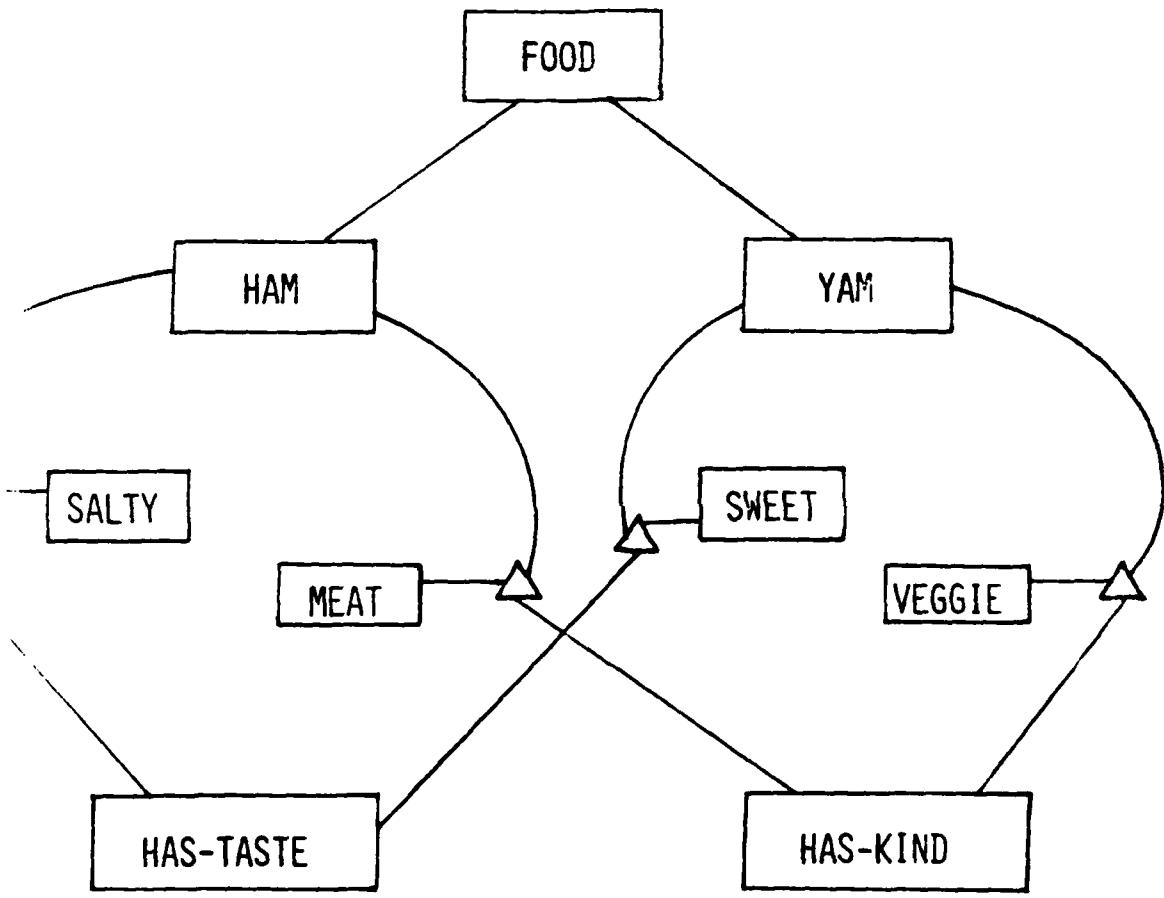
FIGURE 1.4

2. Structure of Knowledge

In developing a framework for representing conceptual knowledge we will concern ourselves with the *internal* representation that a cognitive agent may have of the external world. We would like to emphasize that we are not interested in the external world per se but rather in an agent's conceptualization of the external world. The knowledge embodied in the internal representation of an agent is highly *structured* and *interrelated* and is less a collection of "facts" than an intricately woven fabric of interrelated bits and pieces which fit together to form a *conceptual structure*. In this section we will develop a vocabulary to describe the conceptual structure and to specify the interactions that occur within it. This paper does not address all the relevant issues and although some important problems are analyzed in depth some others are merely identified.The framework shares features with other semantic network and frame based schemes [Bobrow & Winograd 76; Brachman 77; Fahlman 79; Fox 82; Minsky 75; Roberts & Goldstein 77], but differs from all of these in several fundamental respects.

2.1 Conceptual attributes

A cognitive agent interprets the external world in terms of certain *conceptual attributes* and their *values* and all of the agent's world knowledge is represented using these attributes and values. In the restricted context of vision a conceptual structure may be defined in terms of visual attributes like "color" (with values such as red, blue, purple), "shape", "size", "texture" etc. Such a conceptual structure may be extended by including non visual attributes like "weight," "temperature," "odor," "location," "utility" and "function" (i.e. use). In addition to the conceptual attributes mentioned above, typical semantic network relations such as *is-a-kind-of, is-a-part-of, is-an-element-of* are also considered to be conceptual attributes in the proposed framework. The distinction between different kinds of conceptual attributes is discussed in Section 2.3.1.

The explicit identification of conceptual attributes and their values is a crucial step in extracting the structure of knowledge because all other components of the conceptual structure are defined in terms of the conceptual attributes and their values.

Conceptual attributes need not be primitive. For example, a complex conceptual attribute like "shape" may have finer structure consisting of several conceptual "sub" attributes such as "length to breadth ratio" or "relations between sub-parts". Similarly, "physical property" may be regarded as a conceptual attribute in some domain but may be composed of more specific conceptual attributes like "size", "weight", "color" etc.

2.2 Conceptual Entities

The primary level of organization in the conceptual structure centers around the notion of *conceptual entities*. These are *labelled* collections of coherent <conceptual attribute, value> pairs. For instance, an entity labelled FIDO may partially consist of the following <attribute, value> pairs: *"is-an-instance-of* DOG, *is-an-instance-of*

ANIMAL. *has-body-part* LEGS, *has-body-part* TAIL, *has-coat-type* FURRY...". The values of conceptual attributes are conceptual entities and hence conceptual entities may be arbitrarily complex. This definition does not suggest circularity because some conceptual entities are grounded in perception. Conceptual entities are like **concepts** in semantic networks and will often be refered to as such, furthermore, we will often drop the prefix conceptual in conceptual attributes and conceptual entities and refer to these as attributes and entities respectively.

Conceptual entities may denote different sorts of things n the domain such as objects, categories, events, locations and relations. For instance, conceptual entities may denote "my dog Fido", "the color red", "Dog", "Color", "the Sox Phillies game", "the concert tonight" or "John's passing of the ball to Leo".

Different classes of conceptual entities may have different sorts of <attribute, value> pairs associated with them. Thus, physical objects may have attributes mentioned earlier like "*is-an-instance-of*", "*has-color*", "*has-shape*" and "*has-size*" whereas attributes associated with events may be "*has-location*", "*has-agent*", "*has-time-of-occurrence*" etc.

Since values of all attributes are conceptual entities, values of attributes such as *location* and *time* are also conceptual entities. Thus, locations like "Harvard Square," "on the table," "between London and Paris," "in my backyard," are all conceptual entities and so are time specifications like "5 p.m. today," "tomorrow," "before dinner" and "15th November 1983."

Relations constitute a major class of conceptual entities. The representation of relations is similar to that of other conceptual entities discussed above. An argument of a relation is analogous to an attribute of an object. Thus, the representation of an *N*-ary relation is like the representation of an object which has N attributes associated with it. For example, the the two place spatial relation *on* may be characterized as a conceptual entity with two "conceptual attributes": *on-top* and *on-bottom*.

In this formulation, relations *are not primary conceptual entities*. The information encoded in relations is expressible as <attribute, value> pairs of the arguments of the relation. For example, the relation *on* represents information about the *location* of both its arguments and this information may be encoded in the values of the conceptual feature *location* of the arguments. Among other things, relations provide an alternative means of structuring knowledge and may be viewed as *"inversions"* in that they provide a way of expressing information which is not object centered. The inverted representations make it easier to perform certain inferences.

2.2.1 What distinguishes a conceptual entity

Although any possible collection of <conceptual attribute, value> pairs is a potential conceptual entity, *only explicitly labelled collections are conceptual entities*. Which collections of <conceptual attribute, value> pairs will be grouped together to form conceptual entities will depend on the domain being modelled and more importantly, by an underlying *theory of learning or concept formation*. We will say more on this in Section 5.4.

2.3 Conceptual Structure

2.3.1 Properties and Structural Links

Conceptual entities were described above as collections of <conceptual attribute, value> pairs wherein each <conceptual attribute, value> pair related the conceptual entity being described to another conceptual entity. Conceptual attributes are classified into two broad categories: *PROPERTIES* and *structural links*. This distinction is crucial and forms the basis of controlled interactions that may occur between conceptual entities. Italicized lower case will be used to refer to structure link names and italicized uppercase to refer to property names.

Structural links provide the coupling between structure and inference. They reflect the epistemological belief that world knowledge is highly organized and that much of this structure can be factored out to provide general domain independent inference rules. Structural links are attributes that have this quality and are used to provide built-in inference paths. The most representative structural link is the *is-an-instance-of* link that is used for "inheritance" in semantic networks. Our formulation employs an extended notion of property inheritance and includes other structural links such as the *is-a-part-of* links used to infer values of attributes such as *HAS-LOCATION*, and *occurred-during* links [Allen 83] used to make inferences pertaining to time. Each structural link has an associated set of properties that may be inherited along the link and this information is used to perform inferences.

Properties correspond to the intrinsic features of Concepts and may vary from domain to domain. When describing physical objects the relevant properties may be *HAS-WEIGHT, HAS-SHAPE* and *HAS-COLOR*, while events may have properties like *HAS-LOCATION HAS-AGENT* and *HAS-TIME-OF-OCCURRENCE*. Properties roughly correspond to the notion of "roles" of KL-ONE [Brachman 77], "role nodes" of NETL [Fahlman 79] and "slots" of FRL [Roberts & Goldstein 77].

2.3.2 Types and Tokens

A Conceptual entity may be classified as either a **Type** or a **Token**. Elements of the physical world that are interpreted as *instances* by the agent are represented as Tokens in his conceptual structure. For example, Tokens may represent: "Fido the Dog", "the table in my office" and "the location that is the top of my table". On the other hand, Types refer to *abstractions* defined over Tokens. A Type when instantiated or individuated maps into a Token but by itself it does not represent an instance in the external world. Types are summary descriptions that may be viewed as encoding the agent's belief that there are objects in the physical world that *conform* to these description and that these descriptions may be used to make inferences about objects. Examples of Types are "Apple" and "Dog". Types serve two important purposes. They help structure and organize the knowledge about Tokens so that the "quantum" of knowledge remains within manageable bounds and more importantly, they provide the basis for inductive learning and the encoding of abstractions.

The *is-an-instance-of* relation expresses the relationship between Tokens and Types while the inverse relationship between Types and Tokens is expressed by the

"*is-instantiated-by*". Thus, FIDO *is-an-instance-of* DOG and DOG *is-instantiated by* FIDO. Henceforth, we will user the upper case to denote Tokens and Types.

Contrary to many standard interpretations, our framework does not define a Type to be a set of Tokens. Both Types and Tokens are conceptual entities and hence have a similar syntactic structure namely a collection of <conceptual attribute, value> pairs. One difference is that *a value owned by a Type describes the value of a property shared by a large number of its Tokens.* (This notion will be refined in due course). Thus, the Type ELEPHANT may own the *value* GRAY for the *property* HAS-COLOR to represent the fact "most elephants are gray". However, the Type ELEPHANT may not own any value for the property *HAS-AGE* because the instances of elephants may not have any characteristic value for this property.

2.3.3 Hierarchies

The process of abstraction need not stop at one level. Abstractions over Types may yield more abstract Types (or a Type may be differentiated to produce more refined Types). This leads to a hierarchical structure. In general, multiple hierarchies may be defined over the same set of Tokens. For example, a dog is a kind of animal but is also a kind of pet, similarly, a friend of mine is human, a graduate student, male, a classical music aficionado and of course a friend. The result of having multiple hierarchies is that each Token may be related to more than one Type via the same structural relation. The formulation developed here allows any number of hierarchies to be defined on the underlying tokens as long as the resulting structure is acyclic. An example of such a acyclic structure is shown in Figure 2.1. Note that DOG is related to two Types via the *is-an-instance-of* relation namely, ANIMAL and PET. Furthermore, the formulation allows redundant links thus, besides links encoding "DUSTY *is-an-instance-of* GOLDEN-RETRIEVER", and "a GOLDEN-RETRIEVER *is-instantiated-by* DOG", there is also a link encoding "DUSTY *is-an-instance-of* DOG". Redundant links play an important role and the motivation for including them is explained in [Shastri 84].

The hierarchies are not limited to those defined by the relation *is-an-instance-of*. Other structural relations like *is-a-part-of* and *is-a-member-of* also define hierarchies over the tokens. These hierarchies differ in the nature of inheritance that may occur along them. This issue was addressed earlier in Section 2.3.1.

2.4 A representational notation

We will employ a graphical notation in order to present the role of evidence in the representational framework. Figure 2.2 displays a sample network encoding the following information: "Fruits are a kind of Things. Apple is a kind of Fruit. Things have the property color. Apples are generally Red or Green and Red and Green are instances of Color". Arcs in the figure are simplified representations of links; Figure 2.3 shows some of the links in greater detail. There exists a simple mapping between networks depicted in this section and the actual connectionist network implementation is described in Section 4.

The representation uses three kinds of nodes: the Type node, the Token node and the Binder node. Type and Token nodes label collections of <attribute, value>

pairs. While properties and values are accociated to Concepts via Binder nodes, structural links are encoded directly as links. The framework permits associating properties as well as property values with concepts. For example, the binder node b1 in the above network associates the property of having color with Things (and hence Fruits and Apples) without specifying any particular color values. On the other hand the Binder node b2 represents "the value of the property color for Apples may be Red" and also "something that has color Red may be an Apple". The interpretation of b3 is analogous to that of b2 with Green replacing Red. The interpretation of node b4 is slightly different. It represents the *uncertainty* in the system's belief that "Apples can *only* be Red or Green". The exact interpretation of the notation requires taking into account the weights associated with links.

A weight - in the range of 0.00 and 1.00 - is associated with each link and provides the basis for an evidential semantics of knowledge. With reference to Figure 2.2 the weight W1 on the link from b2 to RED is a quantitative measure of the evidence provided by the fact "an object X is an Apple" to the fact "the color of X is Red" and similarly, the weight W4 on the link from b2 to APPLE is a measure of the evidence provided by the fact "the color of an object X is Red" to the fact "X is an Apple". The weights W2 and W5 have a similar interpretation for the relationship between Apple and Green. The above information may be represented in a symbolic notation such as:

$$E(HAS\text{-}COLOR\ RED|\ APPLE) = W1$$
$$E(APPLE|\ HAS\text{-}COLOR\ RED) = W4.$$
$$E(HAS\text{-}COLOR\ GREEN|\ APPLE) = W2$$
$$E(APPLE|\ HAS\text{-}COLOR\ GREEN) = W5.$$

The weights are not independent. For instance, the weights on links from Binders that relate RED to Concepts that are red in color should add up to 1.0 and similarly, the weights on links from Binders relating APPLE to its various color value nodes should also add up to 1.0 (W1 + W2 + W3 = 1.0). The weight W3 is a measure of the "ignorance" or "uncertainty" in the information about Apples and their color and is equal to: 1.0 - (W1 + W2). W6 is equal to W3 but encodes negative evidence for APPLE which comes into play *only if the value specified for color is neither Red nor Green*. If no color value is specified or if the specifed value is Red or Green, the negative evidence is disabled.

We would like to point out some salient features of the representation.

    i. Necessary properties and sufficient properties: The weights on links from owners to binders provide a mechanism for encoding the differences in the strength of the generalizations represented by a Type. Consider the following assertions about Types and properties.

        a) Hexagons have six sides.
        b) Dogs have four legs.
        c) Birds fly.
        d) Apples are red.

a) and b) are examples of assertions with the highest evidential weights, followed by weights of assertions c) and d). In particular, a) is a statement

about a necessary property of hexagons. The use of negative evidence permits the representation of necessary property value. Because a hexagon necessarily has six sides, a) will be encoded as E(HAS-NUMBER-OF-SIDES 6| HEXAGON) = 1.0 and hence HEXAGON would receive an extremely high negative evidence if the object under consideration does not have six sides. The representation also permits representation of sufficient conditions. Imagine that being blue is a sufficient property of blueberries i.e. "if something is blue then there is complete evidence that it is a blueberry" (or equivalently - the only blue things in the domain are blueberries) then this may be represented as E(BLUEBERRIES| HAS-COLOR BLUE) = 1.0.

ii. Multiple property values: In the example discussed above (cf Figure 2.2) Apples could be red or green in color and the representation of the value of the property HAS-COLOR for the Type APPLE accounts for both these colors. If the conceptual structure of the agent is such that red is a more typical color of apples than green, then w1 will be greater than w2.

iii. Distinction between Property and Value: The representation includes a node HAS-COLOR and a node COLOR. These two represent two distinct aspects of the knowledge encoded in the network. The node HAS-COLOR represents a property whereas the node COLOR represents a Type whose instances may include WHITE, BLUE, RED etc, each of which could be a value of HAS-COLOR.

iv. Scope of Properties: Properties are inherited in the same way as property values. Once a property is associated with a Type it gets associated with all Types or Tokens that occur below the Type in the is-an-instance-of hierarchy. Furthermore, a Type or Token may own a value for a property only if the prop is owned by itself or by a Type higher up in the is-an-instance-of hierarchy. For instance, APPLE may own a value for HAS-COLOR because that property is owned by FRUIT which is a superType of APPLE.

2.4.2 Representation of exceptions

In standard representation schemes, attaching a value V1 for the property P1 to a Type T1 is considered equivalent to declaring:

$$\forall x \, TYPE(x, T1) => P1(x, V1),$$

for instance, $\forall x \, TYPE(x, APPLE) => COLOR(x, RED)$

The representation of a Token of T1 which does not agree with the value of the property P1 causes problems. This problem is referred to as the problem of exceptions and cancellation in AI [Fahlman et al. 81; Brachman 82].

There would indeed be a problem if one were to interpret the following two assertions together:

1) $\forall x \, TYPE(x, T1) => P1(x, V1)$
2) $TYPE(A, T1) \& P1(A, V2)$

3)   V1 ≠ V2

There cannot be any satisfactory interpretation of these two statements: they are simply inconsistent. (Assuming $P1(x,y) \Rightarrow (\forall z\ P1(x,z) \Rightarrow z=y)$).

The explicit distinction between Types and properties and the evidential semantics of knowledge provides a natural representation of "exceptions" and gives a clean semantics to the *is-an-instance-of* link. In this framework, *all exceptions are stated in terms of property values* and not Type memberships. Thus, either a Token is an instance of a Type or it is not and the *is-an-instance-of* link states this unequivocally. (The same applies to subTypes and superTypes).

The evidential framework allows the representaion of a range of "quantifications" - universal quantification being a limiting case, and at the same time rules out "genuinely" meaningless statements. In this framework, one cannot both say,

"All Swans are white" and
"Giselle is a Swan whose color is Black".

However, one may say,

"Most Swans are white" and
"Giselle is a Swan whose color is Black."

The following example illustrates the way exceptions are handled:

Let us assume that SWAN is a Type represented in the conceptual structure and that one of the abstractions it encodes is: "all Swans are white" i.e. "if something is a Swan then there is absolute evidence that its color is white". Figure 2.4 depicts SWAN along with an instance HANSA - notice that the weight from b1 to WHITE equals 1.0. If a new instance (Giselle) is introduced such that all its properties match those of SWAN except that it is black, the network modification rules will have to choose between three alternatives: (1) lower the weight W1 thereby redefining SWAN and attach the new instance below SWAN, **or** (2) classify the new instance as something other than SWAN and represent it separately, **or** (3) split SWAN into subTypes on the basis of the property *HAS-COLOR* and attach the instance to the appropriate subType. The first case corresponds to representing exceptions and is illustrated in Figure 2.5. The crucial point is that GISELLE may not be attached as an instance of SWAN unless the weight of the link from b1 to WHITE is reduced to a value less than 1.0.

Not making a distinction between properties and Types leads to very unusual interpretations of the *is-an-instance-of* link (the "infamous" IS-A link). The cancellation of IS-A links to handle exceptions is symptomatic of this confusion. For a discussion of these problems see [Brachman 82].

2.4.3 Types and Prototypes

Use of weighted links leads to an interesting consequence. What would happen if

the network were to "imagine a Type". For a moment assume that this is akin to activating the Type node and letting activation spread to the binder nodes. The weights on the links from owners to binders will select the most typical values of each of the properties owned by the Type (it is reasonable to assume that typical values have higher weights than less typical ones). Thus, the resulting "mental image" of the imagined instance of a Type will correspond to a maximally typical instance of the Type (not necessarily an actual instance). This observation suggests that treating links between owners and binders as weighted links does away with the need of storing exemplars, and prototypes in the representation of Types in order to explain certain behavioral results. The representation of a Type does double duty and acts as if were a prototypical representation besides being an abstract representation of a class of Tokens.

2.5 Evidential Basis of Categorization (Recognition)

In Section 2.4 we had observed that weighted links provide a mechanism for discriminating between different levels of confidence in the generalization made by Types. In this section we consider the obverse namely categorizing an occurrence (assigning a Type to a collection of <property, value> pairs) on the basis of the property values.

It is possible to make the process of categorization entirely symmetrical to the process of retrieval by using the weights on links from binders to owners. With the weights taken into account, a match between the property values of a Token and the property values of a Type can be assigned a metric. This process may be described as that of collecting weighted votes: each Type receives some "evidence" from its binders if a value of the Token's property matches that of the Type's. This evidence is combined and each Type ends up with a quantitative measure for the goodness of its match with the Token. The Type with the highest number wins. Furthermore, the metric can be used to explain what is meant by a Token being a stereotypical or prototypical instance of a category. If a Token has property values that match the most of the typical values of the Type then this Token appears to be more stereotypical. This is a simplified version of the visual categorization model presented in [Feldman 82b]: this paper supplies the elaborated semantic net model promised there. In terms of Rosch's work on prototypicality [Rosch 75; Smith & Medin 81], a Robin matches most of the properties of the representation of the Type Bird whereas a Penguin matches only a few.

2.6 Representation of Relations

Figure 2.6 is an example of how relations are represented. The network encodes the following information:

"ON is a kind of spatial relation
ON has two arguments: the thing on top and the thing at the bottom
A is a ball and B is a cube
A is on B".

Notice that the representation of relations is similar to that of other conceptual entities like APPLE and PEAR. An argument of a relation is analogous to an attribute

of an object. Thus, the representation of the two place relation ON is characterized as a Type with two "properties" (arguments): *on-top* and *on-bottom*.

Many relations such as PARENT-OF and ON either hold or do not hold. However, there are many relations that are best viewed as graded relations. An example of this kind of relation is LIKE; as in "John likes Mary". There are two ways in which a degree of strength may be associated with the representation of such relations. First, "liking" itself may have a degree of strength associated with it; John may "like Mary a lot" or "like her just a little". Second, an agent's belief in the various degrees of John's liking of Mary may also vary. Thus, one may *strongly* believe that "John likes Mary *a little*". We believe that the evidential framework will be suitable for representing such distinctions. We intend to pursue this issue in the near future.
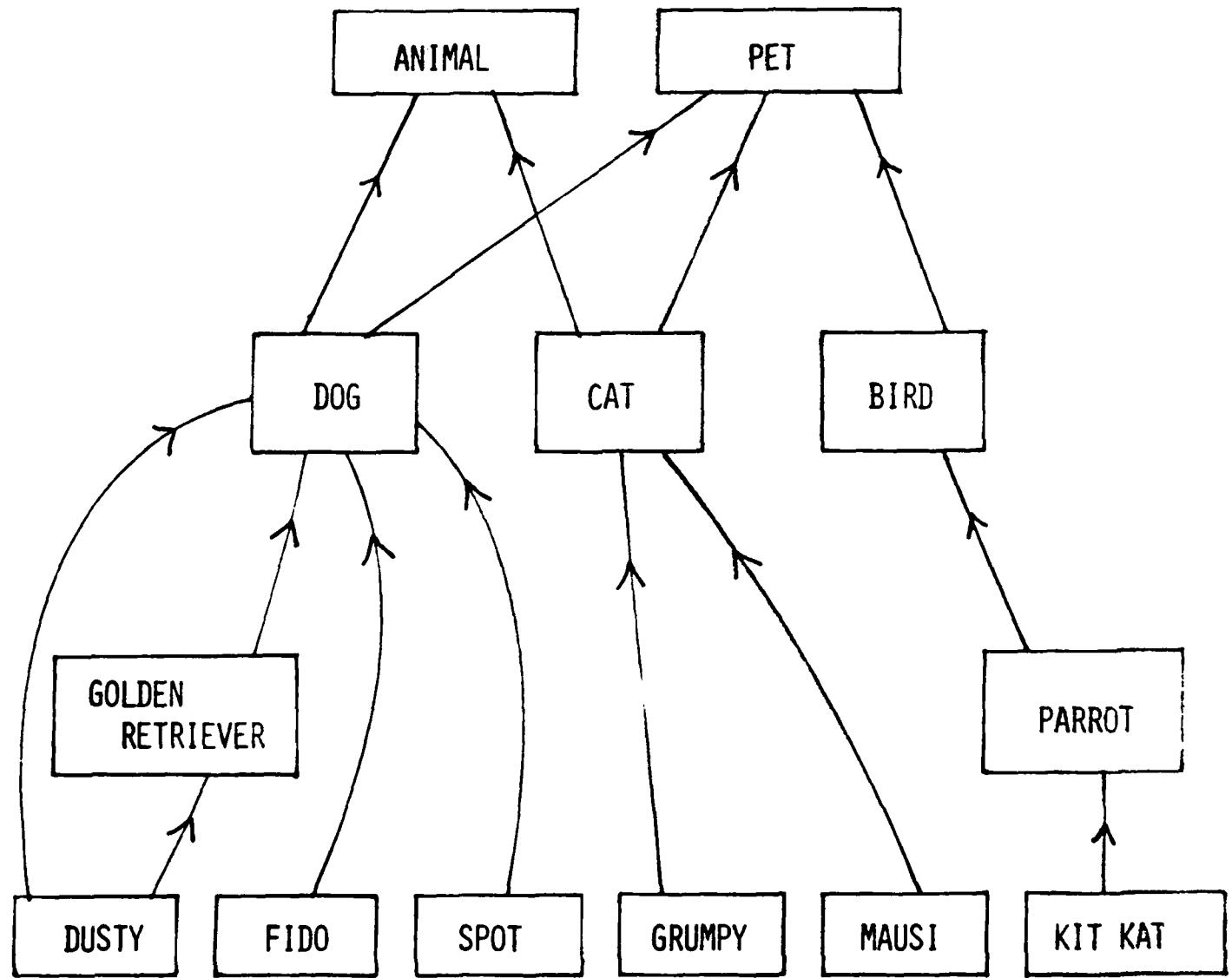
FIGURE 2.1
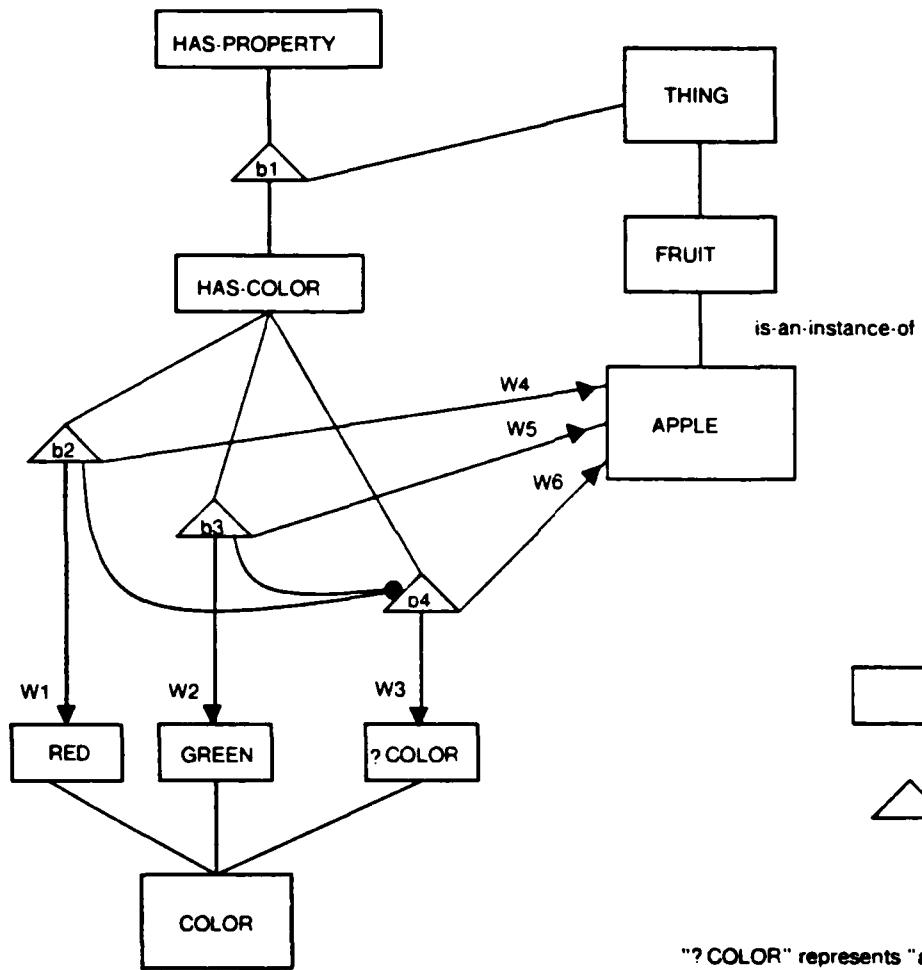
FIGURE 2.2
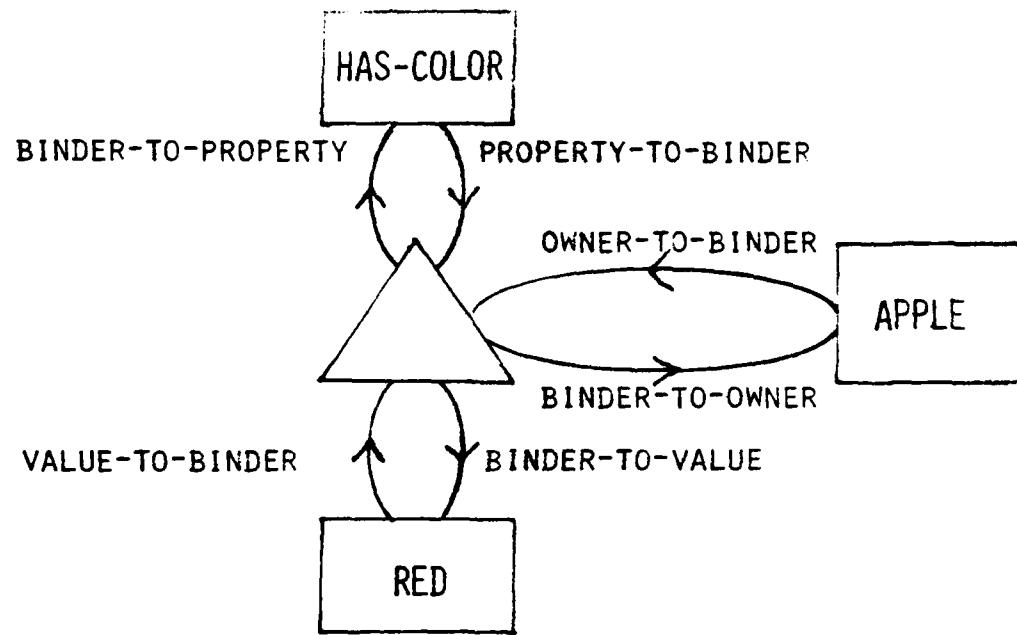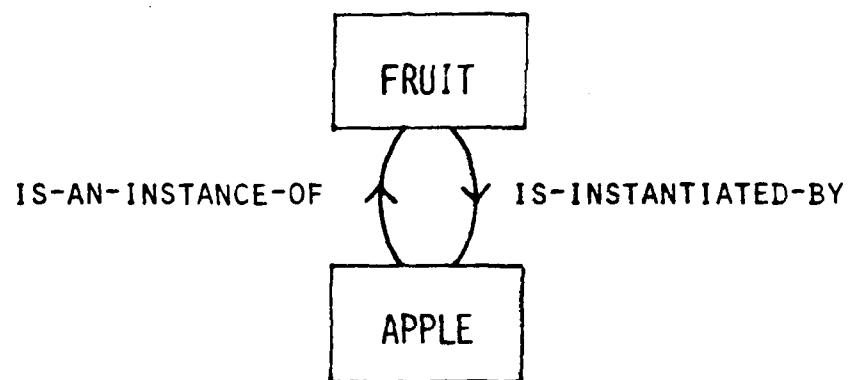
FIGURE 2.3

FIGURE 2.4

w1 < 1 0
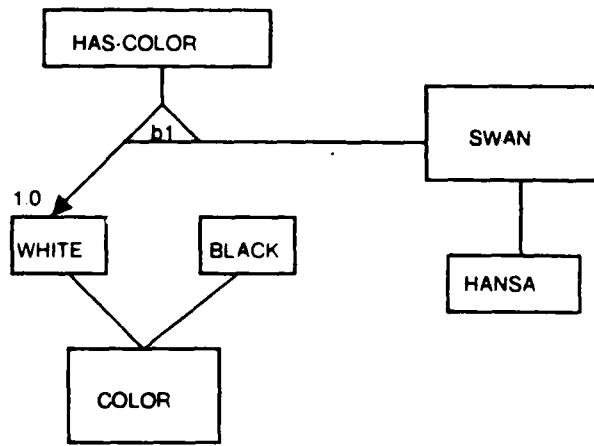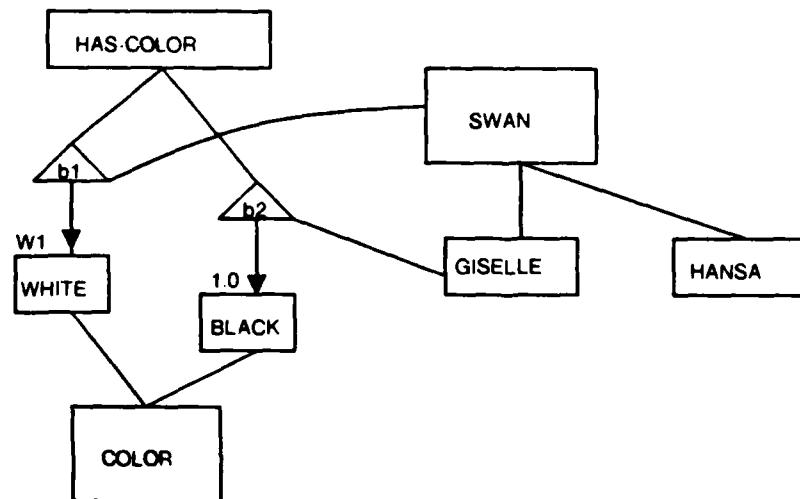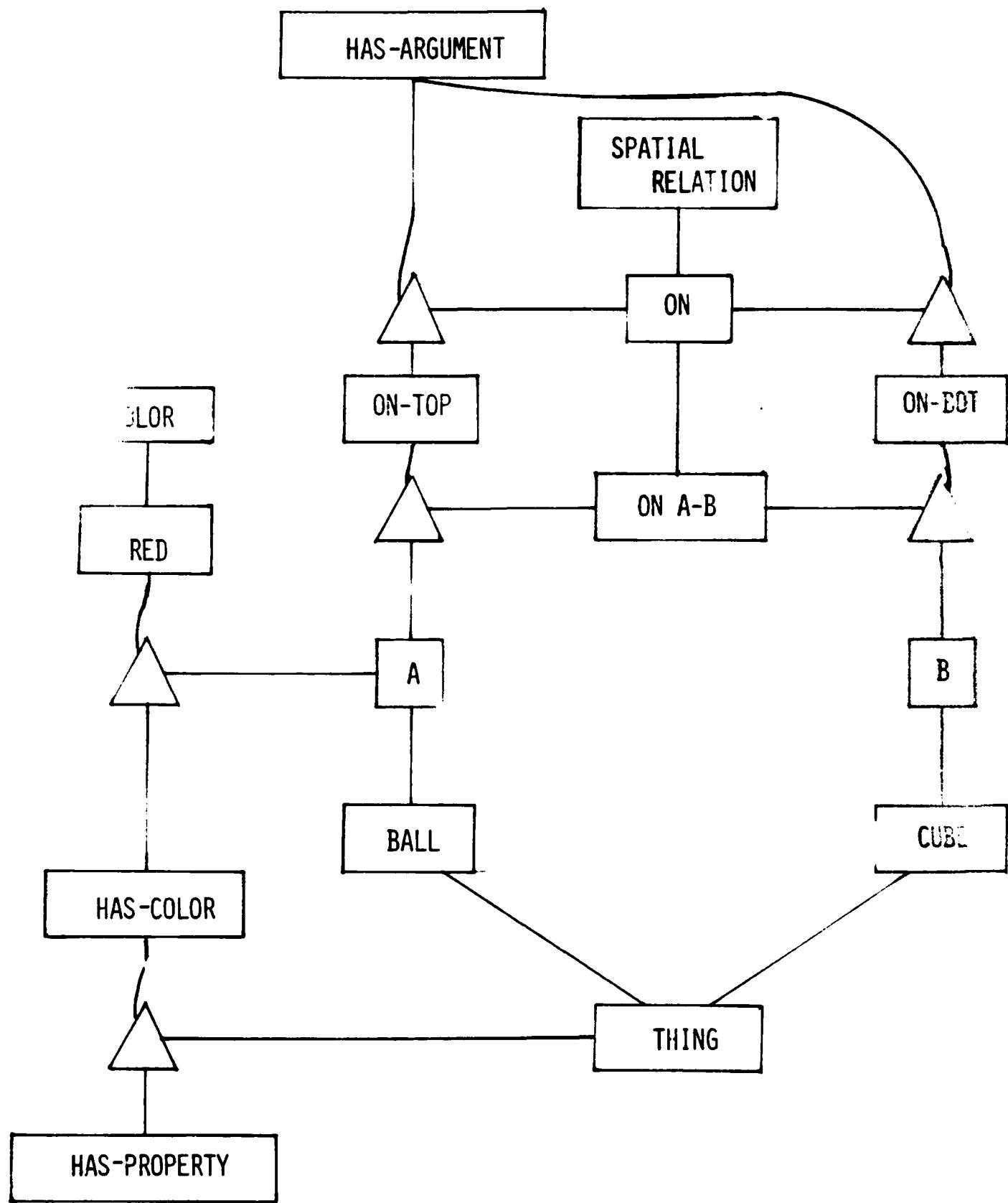
FIGURE 2.5

FIGURE 2.6

3. Inference in memory networks

Section 2 described a notation for representing knowledge and also provided a partial specification of active networks capable of performing limited inferences on the information represented in them. In this section we define the inferences computable within these networks and describe how these can be extended by the use of routines.

Controlled spreading of activation provides the basis for the built-in inference mechanisms in our networks. As was described earlier, a major feature of these networks is that unlike conventional semantic networks, they are not accessed by an interpreter. Consequently, the limited inference mechanisms have to be hardwired into the network and this makes these networks more complicated than typical semantic networks. In order to keep the exposition clear, we will introduce the additional machinery needed to control the spreading of activation as the need arises. The method we have adopted is that of presenting a number of examples that illustrate the various mechanisms involved. The discussion of the exact rules governing the spreading of activation is postponed until Section 4.

As discussed in Section 1, questions for the Memory Network arise from routines and the answers are assumed to be conveyed to Answer Networks which form part of the routines. We will first specify the types of queries the Memory Network can answer and then describe the mechanism for posing a query to and receiving an answer from the Memory Network.

3.1 Nature of Queries

The questions to the Memory Network are framed in terms of conceptual entities (concept), conceptual attributes (attribute) and the values of conceptual attributes. There are three classes of basic queries that may be posed to the network:

Class I Queries:

These queries specify a concept and one of its attributes and seek the value of the specified attribute for the concept. An example of such a query is, "What is the taste of Ham". For the sake of brevity and uniformity we will express all queries of this class as ?v (o a), which may be read as - "What is the value of the attribute a of the owner o. Thus,

"What is the taste of Ham" becomes -
?v (HAM *HAS-TASTE*).

"What is the color of an Apple" maps to -

?v (APPLE *HAS-COLOR*)

As another example, consider the query:

"What is the nose a part of", which is expressed in this notation as,

?v (NOSE *is-a-part-of*).

Class II Queries:

These queries specify one or more attributes along with their values and seek an entity that best matches this description. In our notation these queries are expressed as ?o { (a v) }, which may be read as - "Which object o is described by the following attribute value pairs {(a v)}. Some examples follow:

"Find a red fruit" ->
    ?o { (*is-a-subtype-of* FRUIT) (*HAS-COLOR* RED) }

"Name an animal that flies, is white and quacks" ->
    ?o { (*is-a-kind-of* ANIMAL) (*HAS-MODE-OF-LOCOMOTION* FLYING) (*HAS-COLOR* WHITE) (*HAS-SOUND* "quack-quack") }

Class III queries:

Class III queries seek the attribute that corresponds to an attribute value of an entity. These queries are represented as ?a (o v). For instance,

"What property of Apple has the value Red?" ->
    ?a (APPLE RED)

Queries as multiple choice questions

In this formulation we will assume that all questions posed to the memory network are multiple choice questions. For the purpose of this paper this may be treated as a restriction on the kinds of queries that the network can answer. A wide variety of access to the network essentially consists of dealing with multiple choice questions in the sense that the process of accessing the information in the network may be viewed as selecting the best among a set of hypotheses on the basis of the evidence provided by the network and the query.

Besides including the choices specified in the question, the set of hypothesis being evaluated explicitly includes two additional choices that correspond to the answer "do not know". The additional choices are:

1) "unable to pick a clear winner because of conflicting evidence"
2) "unable to decide because none of the hypotheses is receiving supporting evidence".

The idea of an explicit set of answers fits in well with the routine networks described in section 1 (recall the use of [yes] [no] nodes in the example routine shown in figure 1.1). The use of "do not know" option allows us to explicitly account for uncertainty and is compatible with our evidence theory treatment (Section 5.1). Section 3.4 describes how the "do not know" response may be used for controlling inference. Some situations may require handling questions that are not multiple

choice; such cases are discussed in Section 5.2.

In the light of the assumption that all questions are multiple choice, all queries (at least implicitly) include an enumeration of the possible answers. Thus, the queries corresponding to the above examples would look like:

?v {SALTY SWEET SOUR} (HAM *HAS-TASTE*)

?v {RED BLUE GREEN} (APPLE *HAS-COLOR*)

?v {CHEST TORSO FACE} {(NOSE *is-a-part-of*)}

?o {APPLE PEAR BLUEBERRY} {(*HAS-COLOR* RED)(*is-a-kind-of* FRUIT)}

?o {SWAN DUCK ELEPHANT} {(*is-a-kind-of* ANIMAL)(*HAS-COLOR* WHITE)(*HAS-SOUND* "quack-quack")}

?p {*HAS-TASTE HAS-SHAPE HAS-COLOR*} (APPLE RED).

3.2 Query interface to the memory network

We now describe how routines pose queries to and receive responses from the Memory Network.

Queries originate from hexagonal nodes in routines called Query nodes. Each Query node is connected to the appropriate nodes in the Memory Network. If the routine includes roles that need to be bound during execution, the links between the Query nodes and the appropriate nodes in the Memory Network are established via the Role Network. When a Query node is activated it sends activation to all the nodes it is connected to.

The multiple choices that make up the possible answers to the query are encoded within the routine in the form of a WTA network and are referred to as the Answer Network. These networks contain a node for each of the possible responses to the query and two special purpose nodes called the [?-conflict] node and the [?-no-info] node. The Answer Networks are designed such that the [?-conflict] nodes win the competition if there is a lack of decisive evidence and none of the possible responses is a clear winner while the [?-no-info] nodes dominate the competition if none of the possible responses are supported by the Memory Network.

The overall behavior of the knowledge representation system is a result of the interaction between the nodes in the routine, the Role Network and the Memory Network. Figure 1.3 depicted this interaction crudely, and we will now present it in some detail.

Consider a routine that decides whether some food goes well with red wine. One may imagine such a routine to include the following query: "Which of these best describes the taste of the food: Sweet, Sour or Salty"? The above routine fragment includes a role "food" that gets bound to the appropriate food item when the routine

is invoked. For instance, if the routine were to be invoked to decide "Does Ham go well with red wine"?, the role "food" would get bound to "Ham" via a dynamic link. The connections are depicted in Figure 3.1 which we explain below.

As instantiated in this example, the query encoded in the routine is of the form ?v {SWEET SOUR SALTY} (HAM   HAS-TASTE). This is encoded by *direct* links from the Query node TASTE-OF-FOOD in the routine to *HAS-TASTE*, O-ENABLE and P-ENABLE nodes and via *dynamic* links to the HAM node (the function of the enable nodes is explained below). Following the Query node, the routine includes an Answer Network with a node assigned to each of the three candidate responses. SWEET, SALTY and SOUR (as a matter of convention, we will label these units r-SWEET, r-SALTY, and r-SOUR). The nodes thus assigned are connected, one to one, to the nodes representing the entities SWEET, SALTY and SOUR in the Memory Network. These links are directional and the activation flows from nodes in the Memory Network to nodes in the Answer Network. The latter accumulate activation arriving from the Memory Network and compete with each other to decide on the correct answer. The first node in the Answer Network to cross a preset threshold is considered to be the answer. As explained earlier, the Answer Network also includes the [?-conflict] and [?-no-info] nodes and any of these may dominate under the specified conditions.

3.3 Inference in the Memory Network

We now present examples that illustrate the inference process. The dynamics of these networks and the computational details pertaining to the implementation of these mechanisms are described in Section 4.

Example I

As the first example we consider the query:

?v{SWEET SALTY SOUR} (HAM *HAS-TASTE*)

We have seen the way this query is set up in the networks in Figure 3.1. To see how the query is processed we need to examine the functioning of the triangle shaped binder nodes such as b1 and b2 and the rectangular nodes representing conceptual entities (cf. Figure 3.1).

Each binder node associates an owner with a property and a value. There is a unique binder node for each such triple represented in the Memory Network. Each binder node has three sites named o, p and v which receive inputs from the owner, property and the value respectively. Each site also has an enabling input which must be on for the input to register at the site. The three enable links are called o-enable, p-enable, and v-enable. The enable links are controlled by three global units named O-ENABLE P-ENABLE and V-ENABLE — one for each kind of enable link. On being activated, each of these "global" units turns on all the enable links it controls.

A binder unit is normally in a latent state but becomes active if it receives

coincident activation at two or more of its sites. On becoming active it transmits activation to all the three nodes connected to it. (These are the same nodes that send activation to the binder node). It should be noted that like any other node, the binder nodes also maintain a continuous valued potential which builds up in response to the activation arriving at various sites of the unit and the output of these units is proportional to their potential. Figure 3.2 summarizes the computational behavior of binder nodes. The binder nodes are similar to those proposed in [Hinton 81].

Binder nodes that encode ignorance and provide negative evidence to conceptual entities behave slightly differently. These nodes become active if the enable signal at site v is on and site p is receiving input from the property in presence of the enable signal. On becoming active these nodes send negative evidence to the associated conceptual entities.

Each rectangular node accumulates the activation it receives from other units and saves this value in the form of a potential. It also sends out activation proportional to its potential to all units connected to it. The rectangular units have multiple sites, some of which are mentioned below. Each rectangular unit may have sites for:

inputs it receives from all binder units of which it is the owner,
inputs from all binder units of which it is a value,
inputs from all binder units where it is the property and
inputs from structural links.

The detailed information about enable links and multiple sites will not be displayed in the figures in order to improve readability.

With this introduction we may now describe the steps involved in the processing of the example query: "What is the taste of Ham" i.e.

?v {SALTY SWEET SOUR} (HAM *HAS-TASTE*)

1. The units *HAS-TASTE*, HAM, P-ENABLE and O-ENABLE are activated.

2. The activation spreads and results in the node b1 (cf. Figure 3.1) becoming active as two of its sites - o and p receive simultaneous activation (along with the enable signals).

3. b1 in turn activates SALTY.

4. In the next few time steps, the potential of r-SALTY builds up and as there is no competition it soon reaches a high value indicating that the answer to the query is SALTY. A trace of the potential of selected units is shown in Figure 3.3.

The following five examples are based on the Memory Network shown in Figure 3.4. (The links from the property node *HAS-COLOR* to the various Binder nodes and the enable signals are not shown in the figure). The information encoded in the network may be summarized as follows:

"Apples, Pears and Blueberries are three kinds of Fruits. Apples are generally Red or Green, Pears are generally Green and Blueberries are Blue. Most Red things are Apples, most Green things are Pears but some are also Apples and all Blue things are Blueberries. MAC6 and YEL2 are two instances of Apples".

In terms of the evidential semantics the information encoded is as follows:

E(HAS-COLOR RED| APPLE) = 0.45
E(HAS-COLOR GREEN' APPLE) = 0.25
E(HAS-COLOR ? | APPLE) = 0.30
E(HAS-COLOR GREEN| PEAR) = 0.85
E(HAS-COLOR ? | PEAR) = 0.15
E(HAS-COLOR BLUE| BLUEBERRY) = 0.99
E(HAS-COLOR ? | BLUEBERRY) = 0.01
E(HAS-COLOR YELLOW| YEL2) = 1.0

and

E(APPLE| HAS-COLOR RED) = 0.70
E(APPLE| HAS-COLOR GREEN) = 0.40
E(PEAR| HAS-COLOR GREEN) = 0.60
E(BLUEBERRY| HAS-COLOR BLUE) = 1.0
E(YEL2| HAS-COLOR YELLOW) = 0.50

For each of the following five examples we will state the query, list the nodes in the Memory Network activated by it, specify the structural links that it enables and trace the potential of a select set of nodes.

Example II

Query: ?v {RED GREEN BLUE YELLOW} (MAC6 HAS-COLOR)

Nodes Activated: HAS-COLOR, MAC6, P-ENABLE and O-ENABLE.

Structural Link enabled: is-an-instance-of

Response Nodes: r-RED r-GREEN r-BLUE r-YELLOW [?-conflict] [?-no-info]

Figure 3.5 traces the potential of the nodes: APPLE, the four instances of COLOR and the corresponding nodes in the Answer Network. In brief, the activation moves up the is-an-instance-of link to APPLE. Now both b1 and b2 become active and send activation to RED and GREEN. The stronger evidence for RED results in its dominating GREEN in the Answer Network.

Example III

In this example we demonstrate how information about exceptions plays a role in retrieval from the Memory Network.

Query: ?v {RED GREEN BLUE YELLOW} (YEL2 *HAS-COLOR*)

The only difference in this query and the previous one is that YEL2 is activated instead of MAC6. The potentials of selected nodes is plotted in Figure 3.6.

This example illustrated how the dynamics of the network behavior causes the value of a Token's property to override the value stored at the Type. The computation of exception was affected by two factors: strength of evidence (the higher evidence from YEL2 to YELLOW compared to that from APPLE to RED and GREEN), and the proximity of information (the Binder local to YEL2 became active before the Binders associated with APPLE). In our framework the structuring of knowledge is an integral part of the evidential semantics and neither of these may be treated in isolation. The structuring affects the dynamics of spreading activation and hence the computation of evidence.

Example IV

Query: ?o {APPLE PEAR BLUE-BERRY} {(*HAS-COLOR* BLUE)(*is-an-instance-of* FRUIT)}

Nodes Activated: *HAS-COLOR*, BLUE, FRUIT, P-ENABLE, V-ENABLE

Structural Link enabled: *is-instantiated-by*

Response nodes: r-APPLE r-PEAR r-BLUEBERRY [?-conflict] [?-no-info]

This is an example of a Class II query. All instances of FRUIT receive activation along the *is-instantiated-by* links. BLUEBERRY gets additional evidence from b6 while APPLE and PEAR get negative evidence. Notice that PEAR decays faster than APPLE. This is because the uncertainty about the color of PEAR is less than that about the color of APPLE and hence PEAR receives more negative evidence. The plot of the potentials of the relevant nodes is shown in Figure 3.7.

Example V

Query: ?o {APPLE PEAR BLUE-BERRY} {(*HAS-COLOR* RED)(*is-an-instance-of* FRUIT)}

Nodes Activated: *HAS-COLOR*, RED, FRUIT, P-ENABLE, V-ENABLE

Structural Link enabled: *is-instantiated-by*

Response nodes: r-APPLE r-PEAR r-BLUEBERRY [?-conflict] [?-no-info]

The plot of the potentials of selected nodes is shown in Figure 3.8. The difference between this and the previous example is that the evidence from RED to APPLE was not as strong as that from BLUE to BLUEBERRY. As a result the time response of the network was slower in this example.

Example VI

Query: ?o {APPLE PEAR BLUE-BERRY} {(*HAS-COLOR* GREEN)(*is-an-instance-of* FRUIT)}

Nodes Activated: *HAS-COLOR*, GREEN, FRUIT, P-ENABLE, V-ENABLE

Structural Link enabled: *is-instantiated-by*

Response nodes: r-APPLE r-PEAR r-BLUEBERRY [?-conflict] [?-no-info]

In this example APPLE and PEAR compete with each other because both receive evidence from FRUIT as well as their color value binders. PEAR reaches threshold first because GREEN provides more evidence to PEAR than to APPLE. This is evident from the plot of the potentials of these nodes shown in Figure 3.9. The behavior of r-APPLE and r-PEAR nodes during steps 11 through 15 reflects the competition between the two nodes and the emergence of r-PEAR as the winner.

To illustrate the role of the [?-conflict] node we modify the Memory Network in Figure 3.4 so that the evidential weights on links from GREEN to APPLE and PEAR become relatively similar. Specifically, we set:

$$F(APPLE| \textit{HAS-COLOR} \ GREEN) = 0.45$$
$$\text{and } F(PEAR| \textit{HAS-COLOR} \ GREEN) = 0.55.$$

We now pose the same query to the modified network and trace the behavior of the Answer Network nodes. The potentials of the relevant nodes are plotted in Figure 3.10. The [?-conflict] node gradually gains potential as neither r-PEAR nor r-APPLE are able to dominate.

Example VII

This example demonstrates inheritance along a structural link other than the *is-instantiated-by* link. The Memory Network in Figure 3.11 encodes the following information:

"Events represent a Type of conceptual entity.
Events have the properties, location and time of occurence (besides others....).
I broke my arm during my first year at college.
I entered college in 1974."

Assume that the query is:

"In which year did I break my arm?"

?v{1972 1974 1976} (*HAS-TIME-OF-OCCURRENCE* BREAK-ARM)

The Query Network and the Answer Network might be set up as shown in Figure 3.11.

It was observed in Section 2.3.1 that each structural link is appropriate for the inheritance of certain property values only. An example of this is the inheritance of the value of the property *HAS-TIME-OF-OCCURRENCE*. The value of this property may be inherited by moving up the *occurred-during* links. The routines encoding a query about the the time of occurrence of events will enable the *occurred-during* links if necessary.

The example Query activates the units *HAS-TIME-OF-OCCURRENCE*, "I-broke-arm", P-ENABLE and O-ENABLE. It also enables the *occurred-during* structural link.

The activation moves along the *occurred-during* link and activates "My first year in college" unit. The binder unit b1 becomes active and sends activation to "1974" which in turn supports r-1974. Finally, r-1974 reaches a high degree of activation and this completes the processing of the query.

Example VIII

As the last example of this subsection, we see how the question, "Is Dick a Pacifist" answered by a network which encodes the assertions, "Quakers are Pacifists", "Republicans are non Pacifists", "Dick is a Quaker" and "Dick is a Republican". The Memory Network in Figure 3.12 encodes the above information. The weights w1 and w2 encode the strengths of evidence E(*HAS-BELIEFS* PACIFIST| QUAKER) and E(*HAS-BELIEFS* NON-PACIFIST| REPUBLICAN) respectively. Needless to say, the encoding is an simplification but the claim is that it brings out the flavor of the manner in which such information is encoded.

Query: ?v (non-PACIFIST PACIFIST) (DICK *HAS-BELIEFS)*

Units activated: The Query activates the units DICK. *HAS-BELIEFS* P-ENABLE and O-ENABLE.

Structural Links enabled: *is-an-instance-of.*

Response Units: r-PACIFIST r-non-PACIFIST [?-conflict] [?-no-info]

The activation moves up the *is-an-instance-of* links to QUAKER and REPUBLICAN units and both PACIFIST and non-PACIFIST receive activation in proportion to the weights w1 and w2 respectively.

An example trace with a particular choice of weights is shown in Figure 3.13.

3.4 Routine-based Inferences

The previous subsection showed how a number of different kinds of basic inference could be captured within the connectionist semantic network. By augmenting these mechanisms with routine-based rules of inference, we can greatly extend the class of inferences supportable by the system. Although we will not press any claims at this time, it appears that the current mechanisms can handle any chain of inference that does not require backtracking (cf. Section 5.2).

We have already used routines in several important ways in inferencing. Questions were assumed to arise from, and answer networks to reside in routines. The question nodes provided simultaneous activation to the parameters of the query and to the appropriate ENABLE nodes for binders and for structural links. The role~concept dynamic-link binding network was not stressed, but provides the system with its basic ability to handle variables, subroutines etc. The answer network mechanism can also be extended and we will do this first because it is simple.

The major addition to our previous treatment of answer networks is to account explicitly for indecision. Suppose, in the luncheon script of Figure 1.3, the special is a burrito and our hero doesn't know if he likes them or not. This would lead to domination by the [?-no-info] node in the WTA of the Answer network. The answer [?-no-info] can, like any other answer, lead to subsequent routine actions. Figure 3 depicts a situation where insufficient information leads to the enabling of is-subtype-of links in the case that was not part of the original query. This also suggests how the mechanisms described here can be used to provide varying degrees of control over spreading activation in semantic networks. It turns out that the explicit [?-no-info] node also plays an important role in our evidence theory as discussed in Section 5.1.

Another use of routines is to access the relational knowledge encoded in the semantic network. The example in Figure 3.15 extends the luncheon routine to include a check against one's supper plans before ordering the special. The new node [conflict with supper?] works by simultaneously activating the roles [special] and [supper] and the relation name [not on same day]. This (in a few steps) would cause activation of [instance 6214] which is a positive instance of conflict between foods and thus linked into the r-yes node in the routine. Similar mechanisms will work for any query of the form R(A,B) where any of R, A, B can be variables (roles) bound to particular concepts. Another routine could provide activation to e.g. all foods that shouldn't be eaten after ham by activating [ham], [first], and [not on same day] to get instances active and then activating [second] and [not on same day] to route activation to foods such as [pork]. We have not yet said how one could then make use of this diffuse activation in the network (cf. Section 5.2).

Another question that might arise is how the binding [supper~pork] came about. It could be that the binding remained from morning as a kind of intermediate term memory, but would only handle a restricted set of cases. Figure 3.16 suggests a general way that such a binding might be computed by an embedded (sub)routine. The important point for us is that the [role~concept] mechanism provides a natural way of linking together routines, quite like the binding mechanisms in logic or programming. The (somewhat fanciful) routine in 3.16 has our hero employing the strategy of imagining the situation (cf. [Feldman 82b]) of his kitchen that morning and focussing on the traditional defrosting counter.

This class of problem has not been worked out as carefully as some earlier examples, but the basic ideas are similar. Activation of the appropriate situation node and relation query would lead to activation of the appropriate unit in the network. The difference here is that the "answer" is being used to establish a binding in the role network of supper ~ pork. There would have to be enabling links to facilitate such bindings, but this is straightforward. The general idea is that any route network

can both take and return role ~ concept bindings, significantly broadening the range of inferences computable with our networks.

The use of multiple role~concept bindings in a routine gives rise to a computational problem that is particular to and ubiquitous in connectionist models [Feldman & Ballard 82]. By depending on parallel spreading activation, we become subject to the possiblity of false coincidences or cross-talk producing false responses. This is particularly tricky since a role can be occupied by different concepts, but the knowledge is all encoded by concepts. An example situation is one where John loves Mary, Mary likes John and John has been transferred. We suppose that this event makes John sad and Mary relieved. The technical problem here is that the roles of John and Mary could be reversed and the mechanism must support either binding. The routine fragment of interest simply activates [lover is sad] and [lovee is relieved] sequentially. Each person is assumed to have several affective states including [happy], [sad] and [relieved]. Consider the spread of activation caused by the first action of the script. Activating [lover is sad] causes activity to spread to the [sad] node of everyone, including John and Mary. At the same time, the [lover] role in the dynamic link network is triggered and this causes activation of [John]. This, in turn, activates [John's affect] and all its possible values. Now in the entire SN, there is only one unit -- [sad] of [John's affect] -- which is receiving coincident activation. We assume that this coincident activation raises the potential of John's [sad] unit: this is the mechanism proposed for capturing the sadness of John in our SN.

The second action [lovee is relieved] will, of course, lead to coincident activation of the [relieved] node of Mary. It is important to notice that it is *not* possible to do these two actions in parallel. If both [lover is sad] and [lovee is relieved] were simultaneously activated, then both the [lover] and [lovee] roles would become active. This would lead, for example, to concident activation of the [relieved] node of John as well as to the desired coincidences. This problem is an instance of the general *crosstalk* problem in connectionist networks [Feldman & Ballard 82]. Whenever one uses coincidence for inference, care must be taken to insure that no false coincidences arise. This is most often done by sequential execution of separate steps. The formation of role~concept bindings itself is one such case. At least for our formulation, sequential processing is required whenever bindings are being established (cf. [Anderson 83]). There are undoubtedly many other problems that will arise in connectionist inference models; the current section is mainly intended to lay a framework for a detailed further study of routine inferences.

FIGURE 3.1

PROPERTY

P-ENABLE

TO PROPERTY

O-ENABLE

P

OWNER

O

TO OWNER

V

TO VALUE

V-ENABLE

VALUE
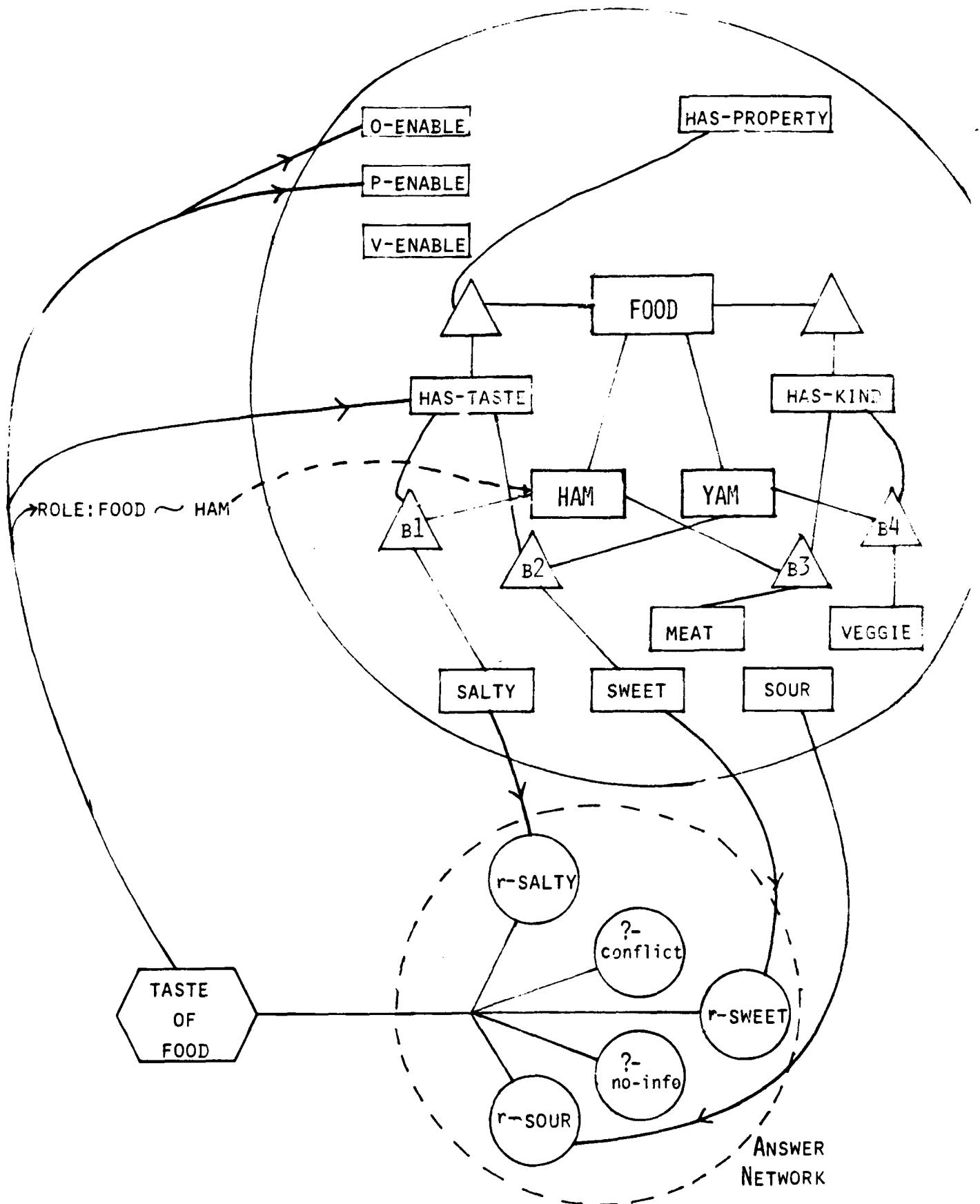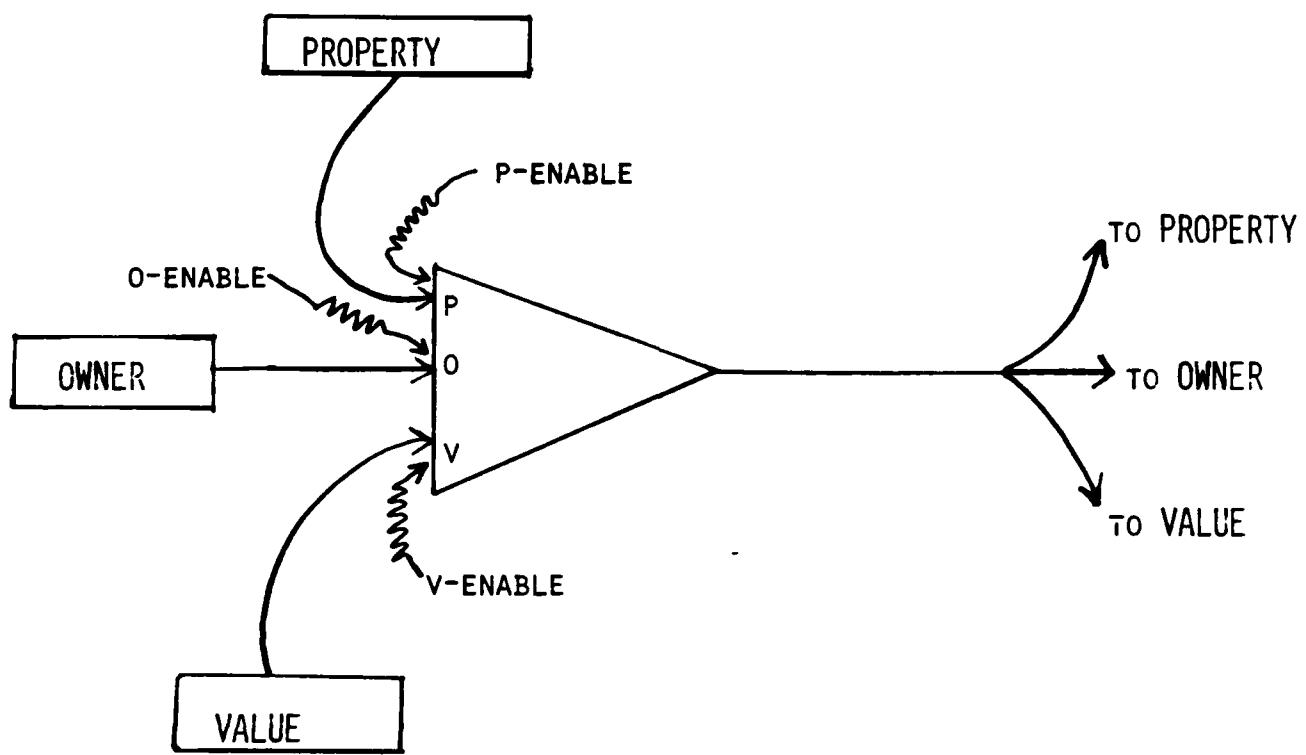
- BINDER UNIT ACTIVE IF TWO OR MORE SITES ENABLED AND RECEIVING INPUT
- UNITS SEND OUTPUT WHEN IN ACTIVE STATE

FIGURE 3.2
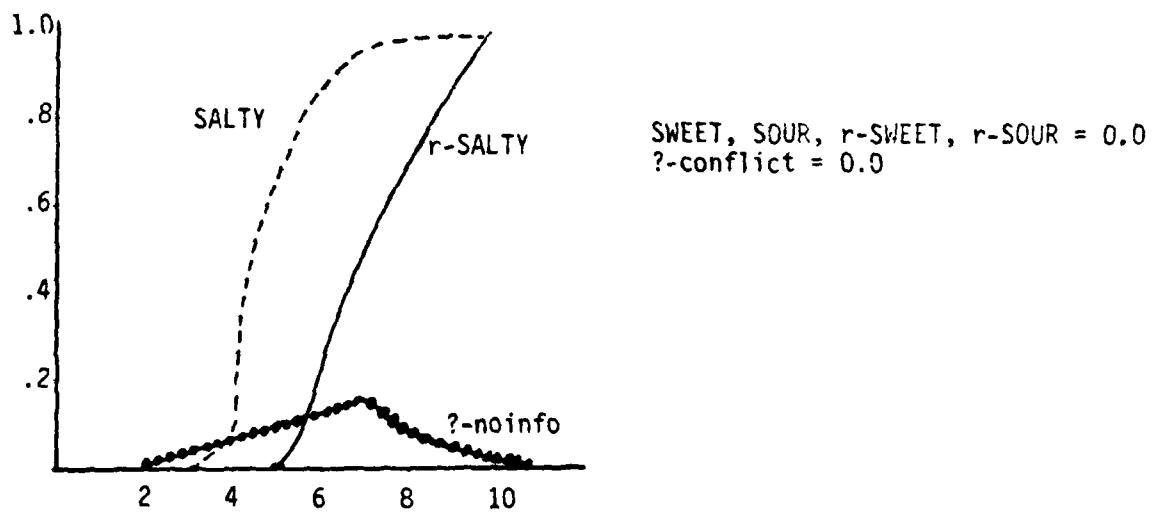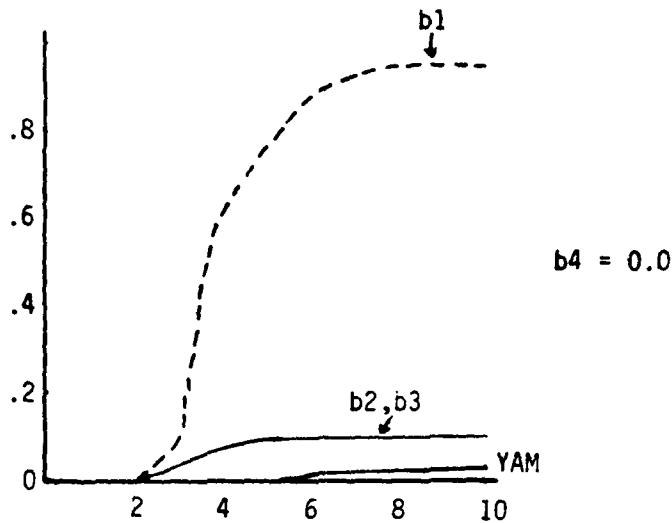
FIGURE 3.3 FIND THE TASTE OF HAM
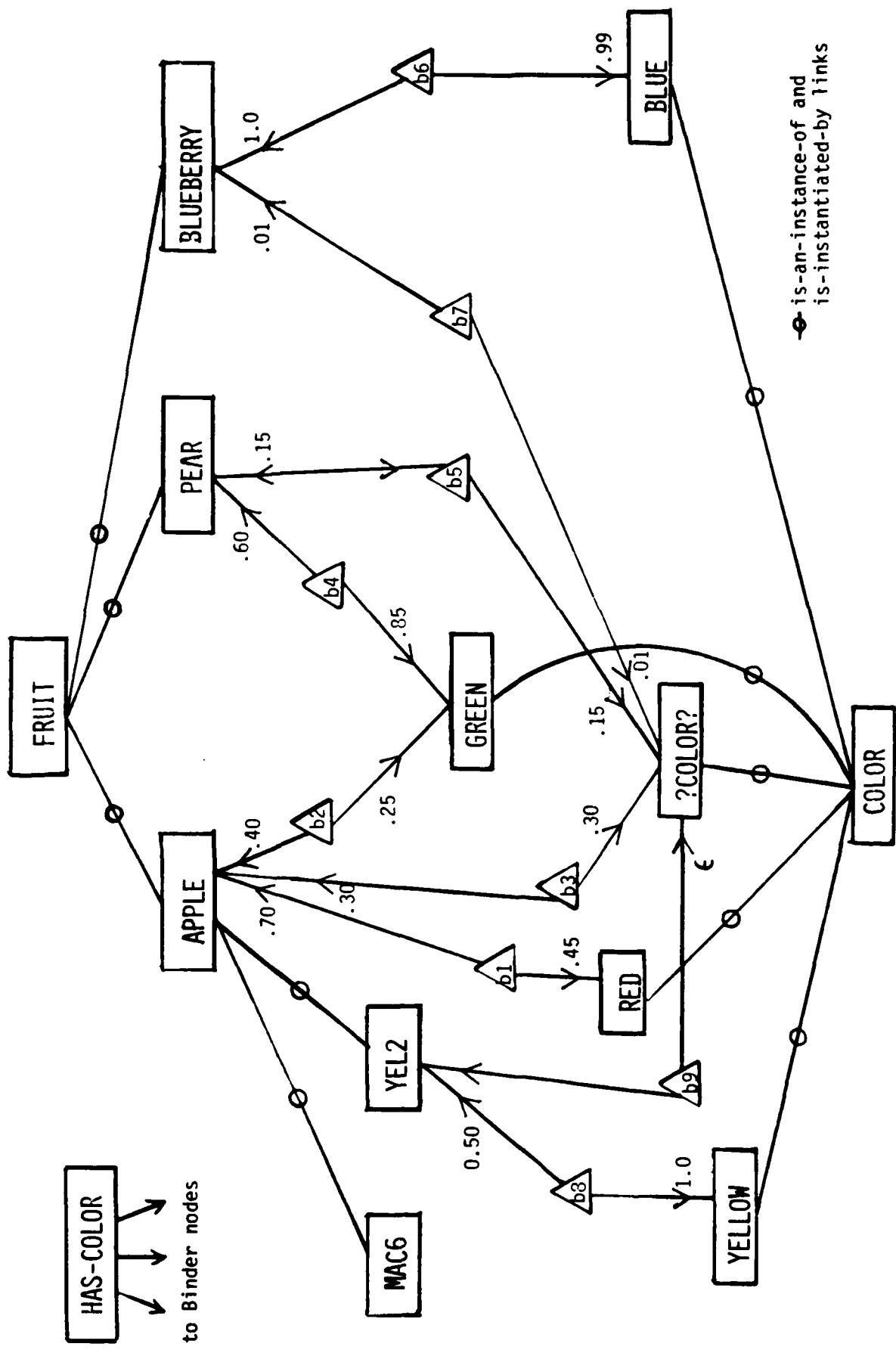
ALL BINDER NODES HAVE AN INPUT FROM HAS-COLOR AND INPUTS FROM ENABLE NODES

FIGURE 3.4
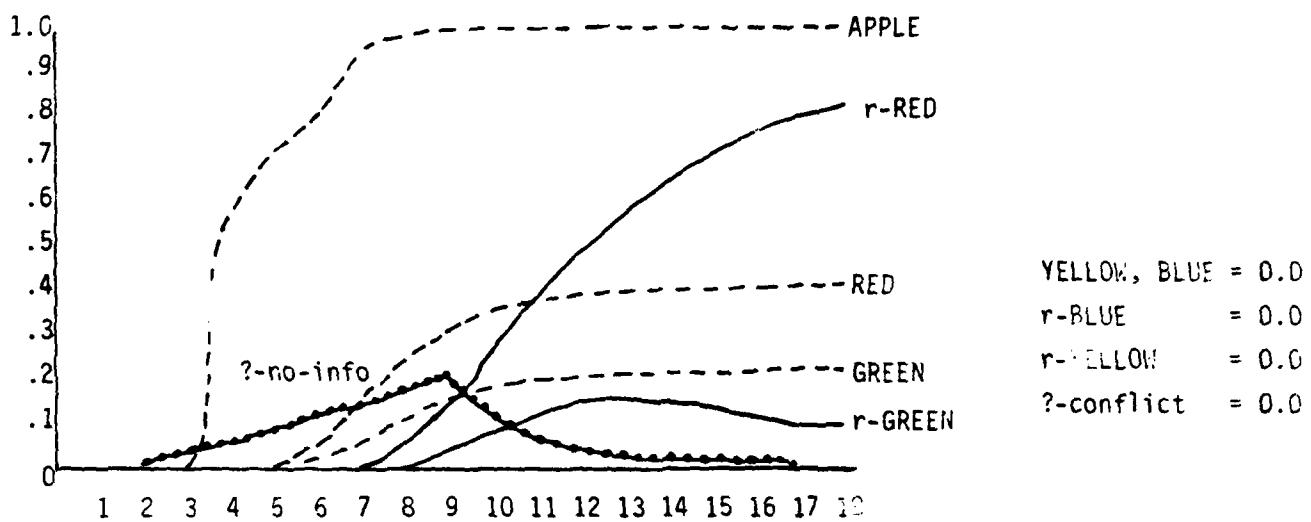
QUERY: ?V (RED GREEN YELLOW BLUE)   (MAC6 HAS-COLOR)

FIGURE 3.5 FIND THE COLOR OF MAC6
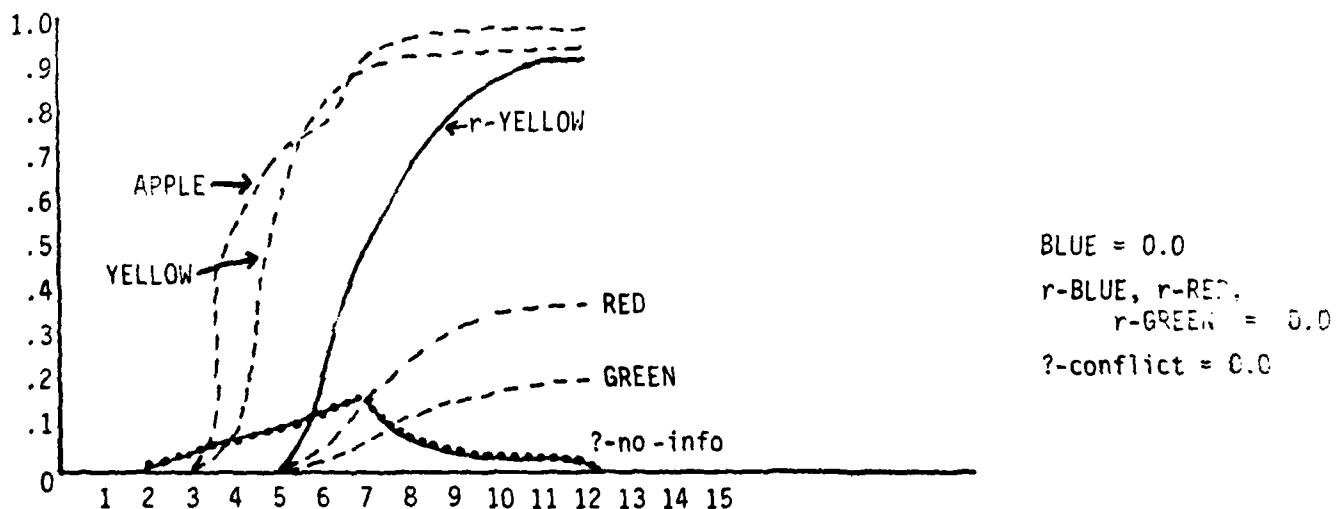
QUERY: ?V (RED GREEN YELLOW BLUE)   (YEL2 HAS-COLOR)

FIGURE 3.6 FIND THE COLOR OF YEL2
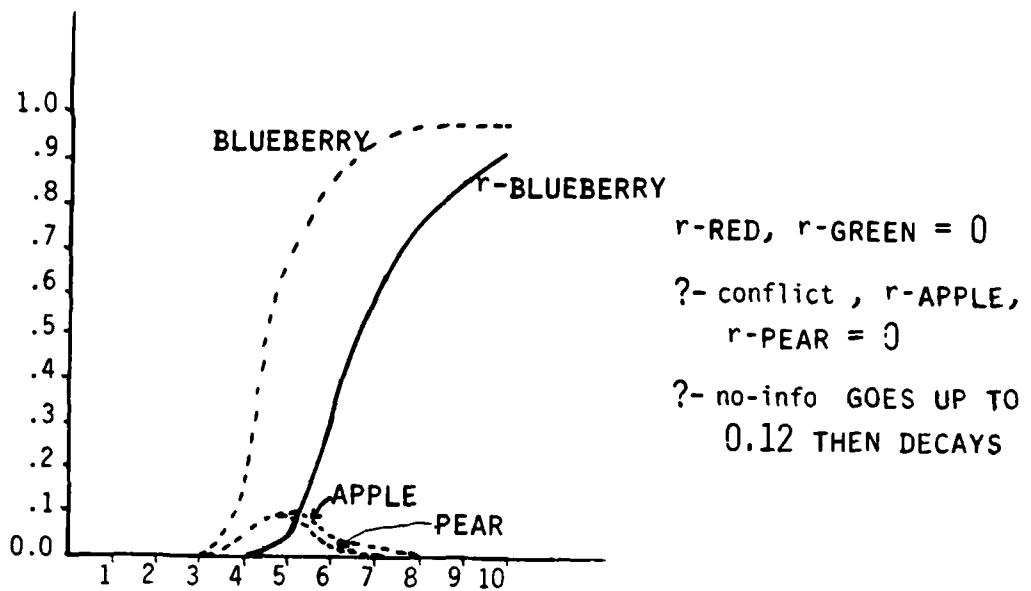
QUERY: ? 0 (APPLE PEAR BLUEBERRY) {(IS-AN-INSTANCE-OF FRUIT)
                                                    (HAS-COLOR BLUE)}

FIGURE 3.7 NAME A BLUE FRUIT
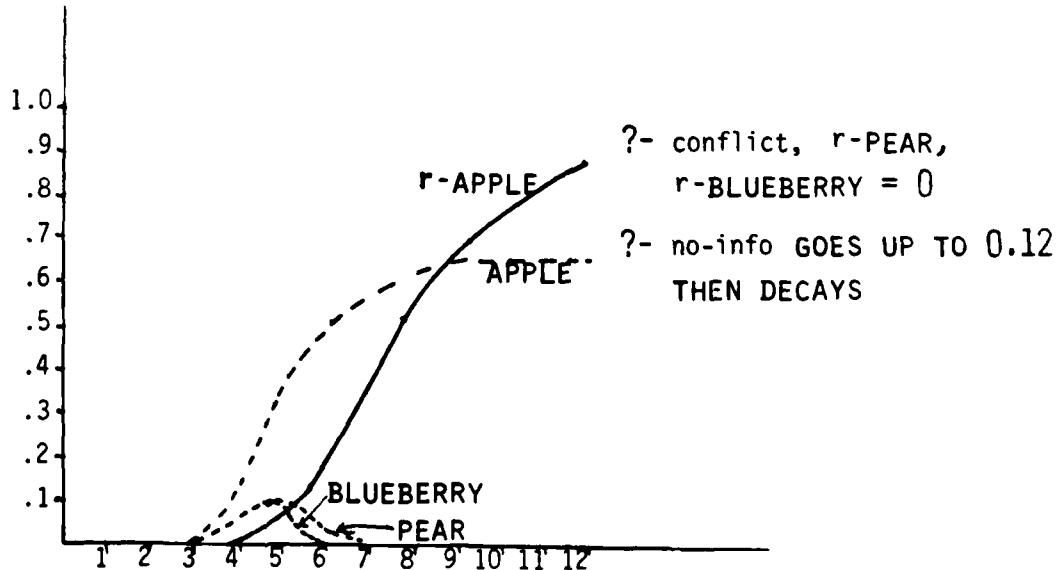
QUERY: ?0 (APPLE PEAR BLUEBERRY) {(IS-AN-INSTANCE OF FRUIT)
                                                  (HAS-COLOR RED)}

FIGURE 3.8 NAME A RED FRUIT
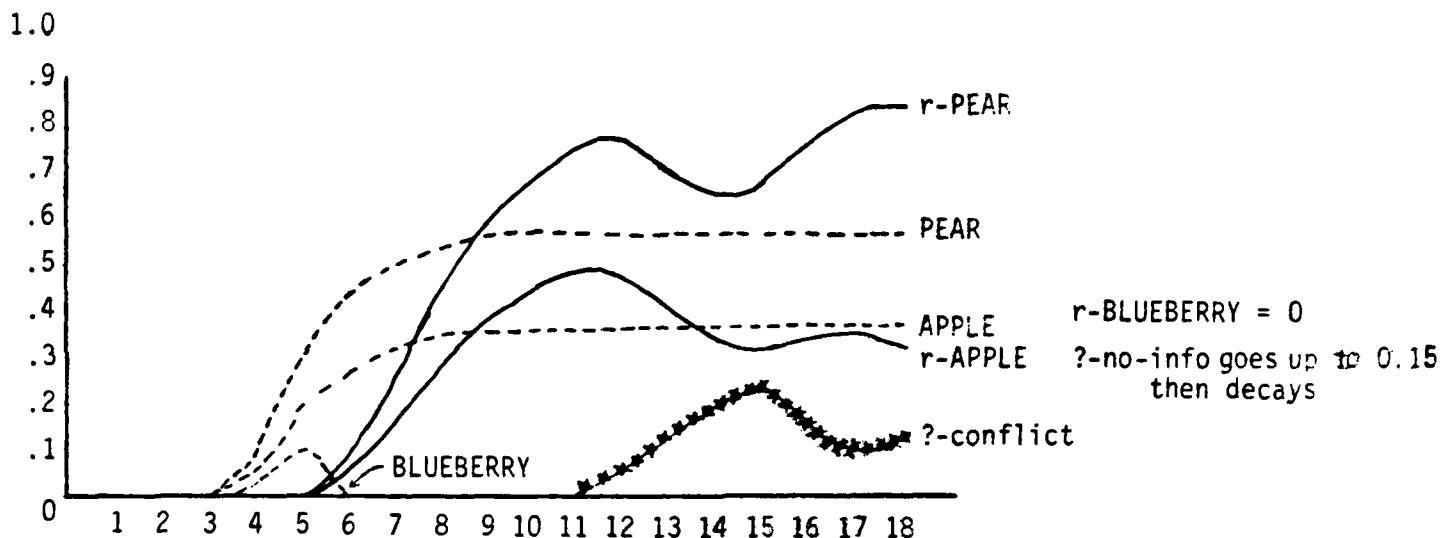
E(PEAR | HAS-COLOR GREEN) = 0.60

E(APPLE | HAS-COLOR GREEN) = 0.40

QUERY   ?0 (APPLE PEAR BLUEBERRY) {(is-an-instance-of FRUIT) (HAS-COLOR GREEN)}

FIGURE 3.9 Name a green fruit

QUERY: ? 0 (APPLE PEAR BLUEBERRY) { ((IS-AN-INSTANCE-OF FRUIT) (HAS-COLOR GREEN))

$$\left\{\begin{array}{l} E(PEAR \mid HAS\text{-}COLOR\ GREEN) = 0.55 \\ E(APPLE \mid HAS\text{-}COLOR\ GREEN) = 0.45 \end{array}\right\}$$

FIGURE 3.10

ENABLE UNITS NOT SHOWN

OCCURRED DURING LINKS ENABLED

FIGURE 3.11

FIGURE 3.12

1.0

.8 ——————————————————— r-PACIFIST

.6

PACIFIST

.4

NON-PACIFIST→ r-NON-PACIFIST

.2

?- no-info ?-conflict

3   5   10   15   20
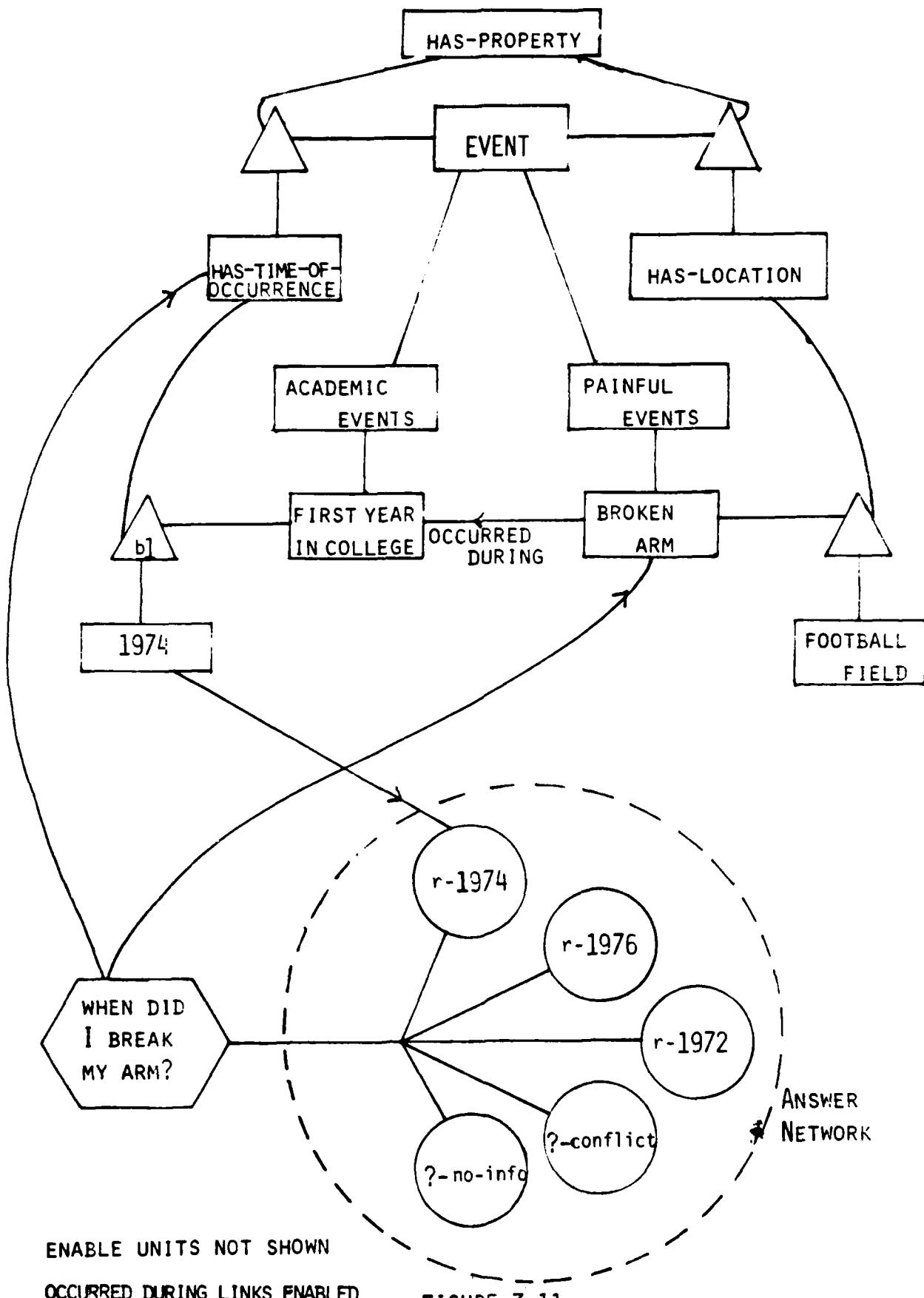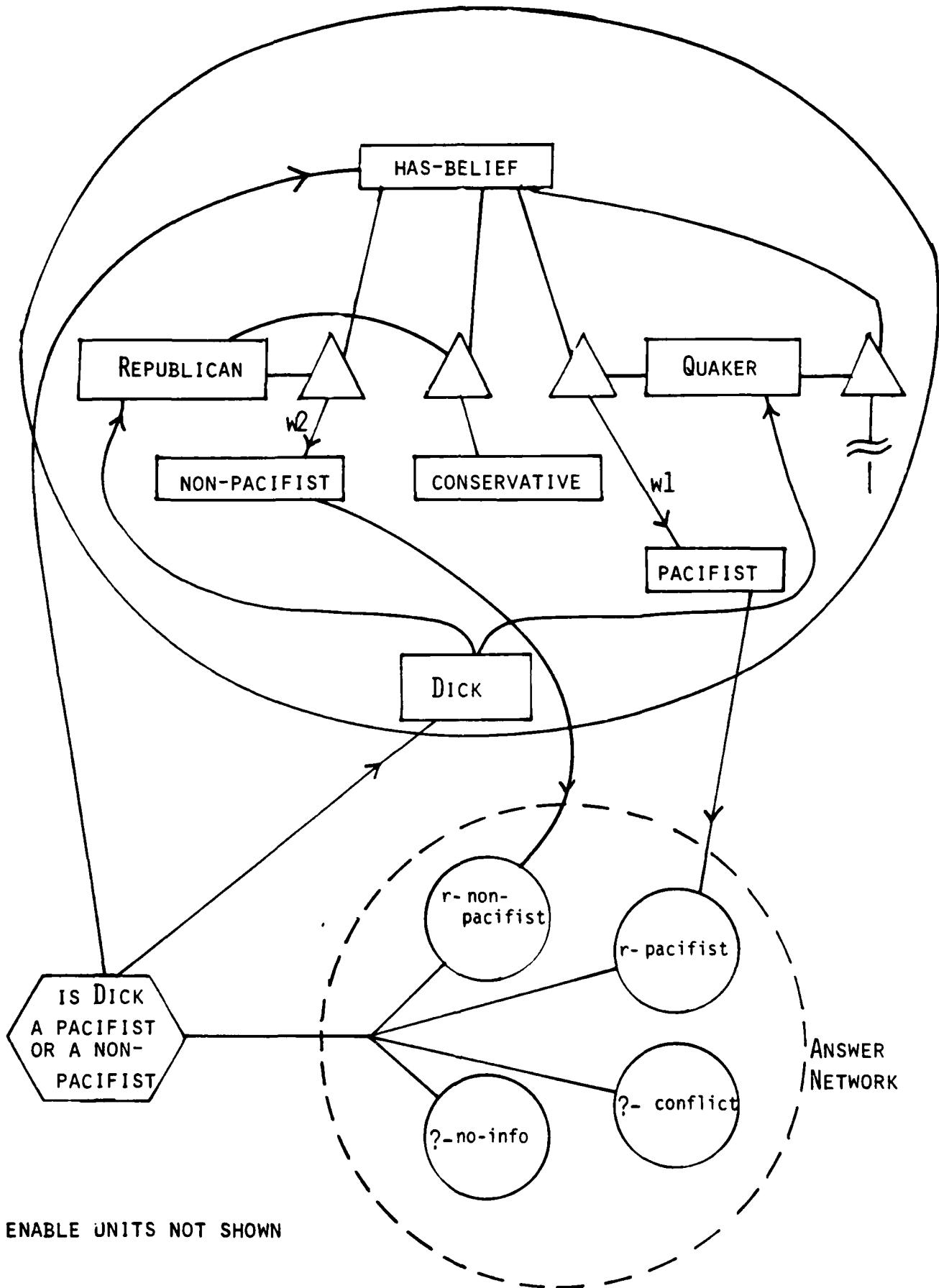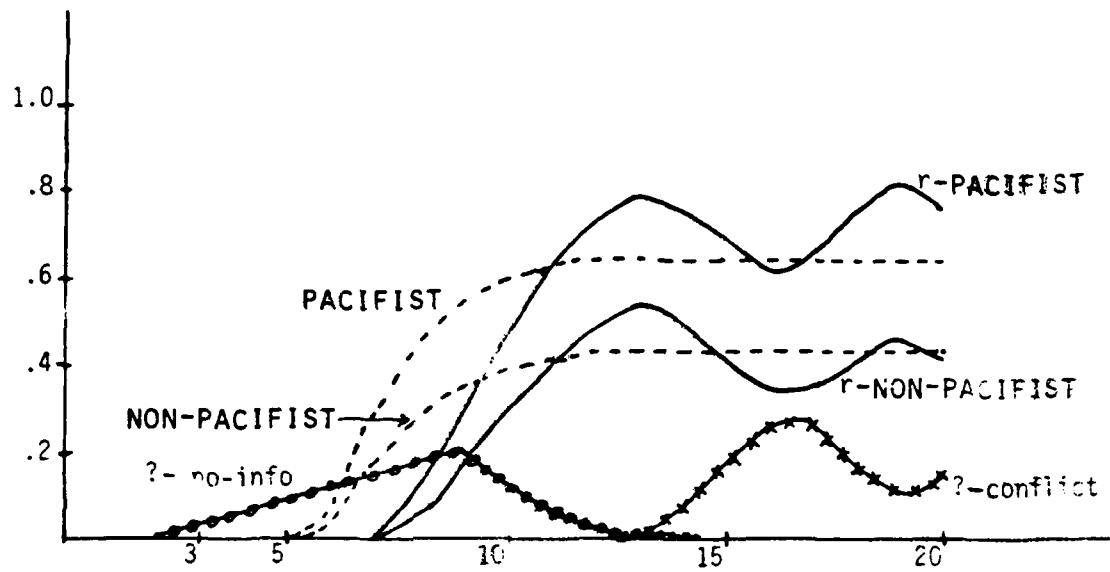
W1 i.e. E(PACIFIST|QUAKER) = 0.7

W2 i.e. E(NON-PACIFIST|REPUBLICAN) = 0.5

QUERY: ? v (PACIFIST NON-PACIFIST) (DICK HAS-BELIEF)

FIGURE 3.13 IS DICK A PACIFIST OR A NON-PACIFIST?
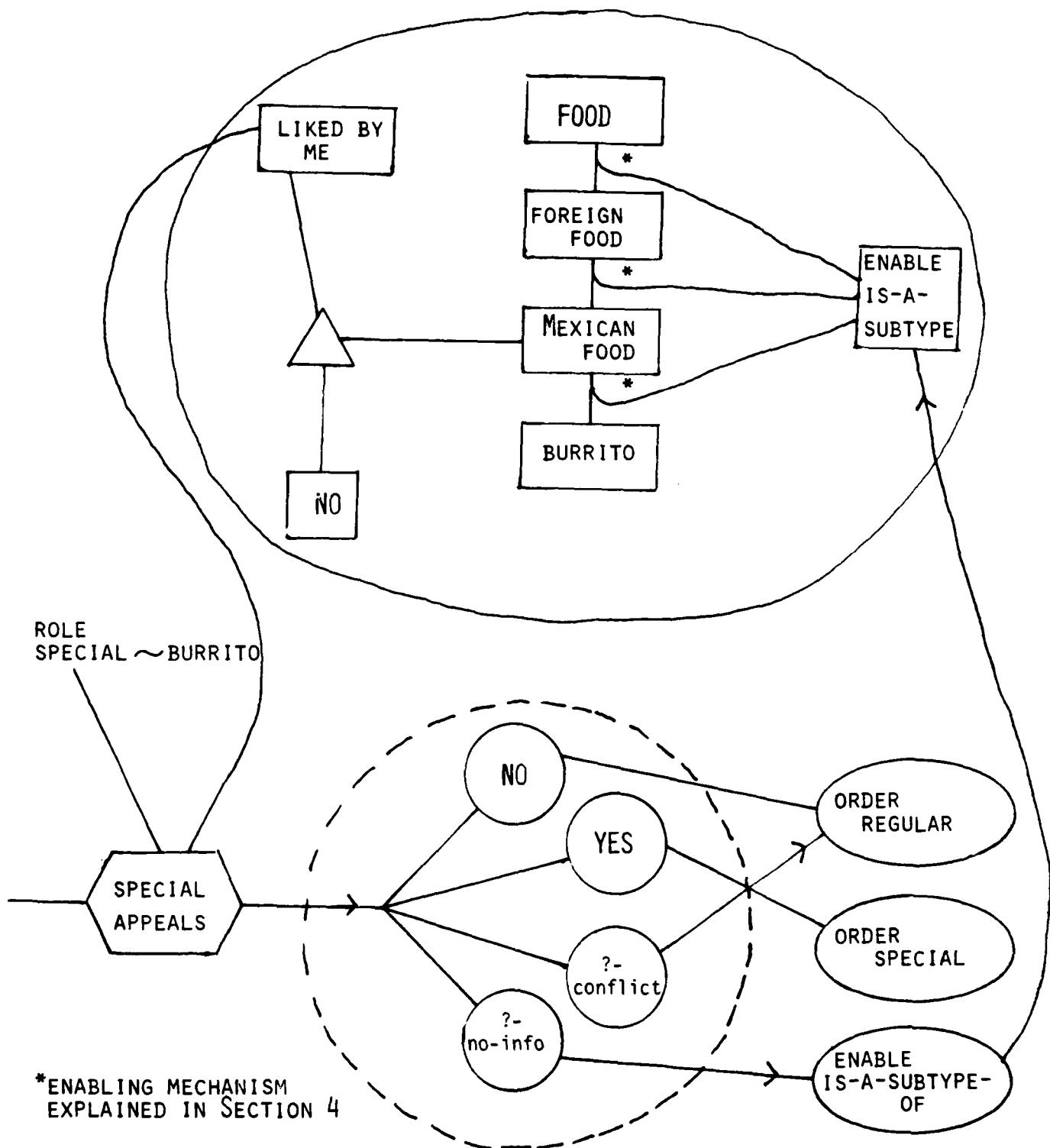
FIGURE 3.14
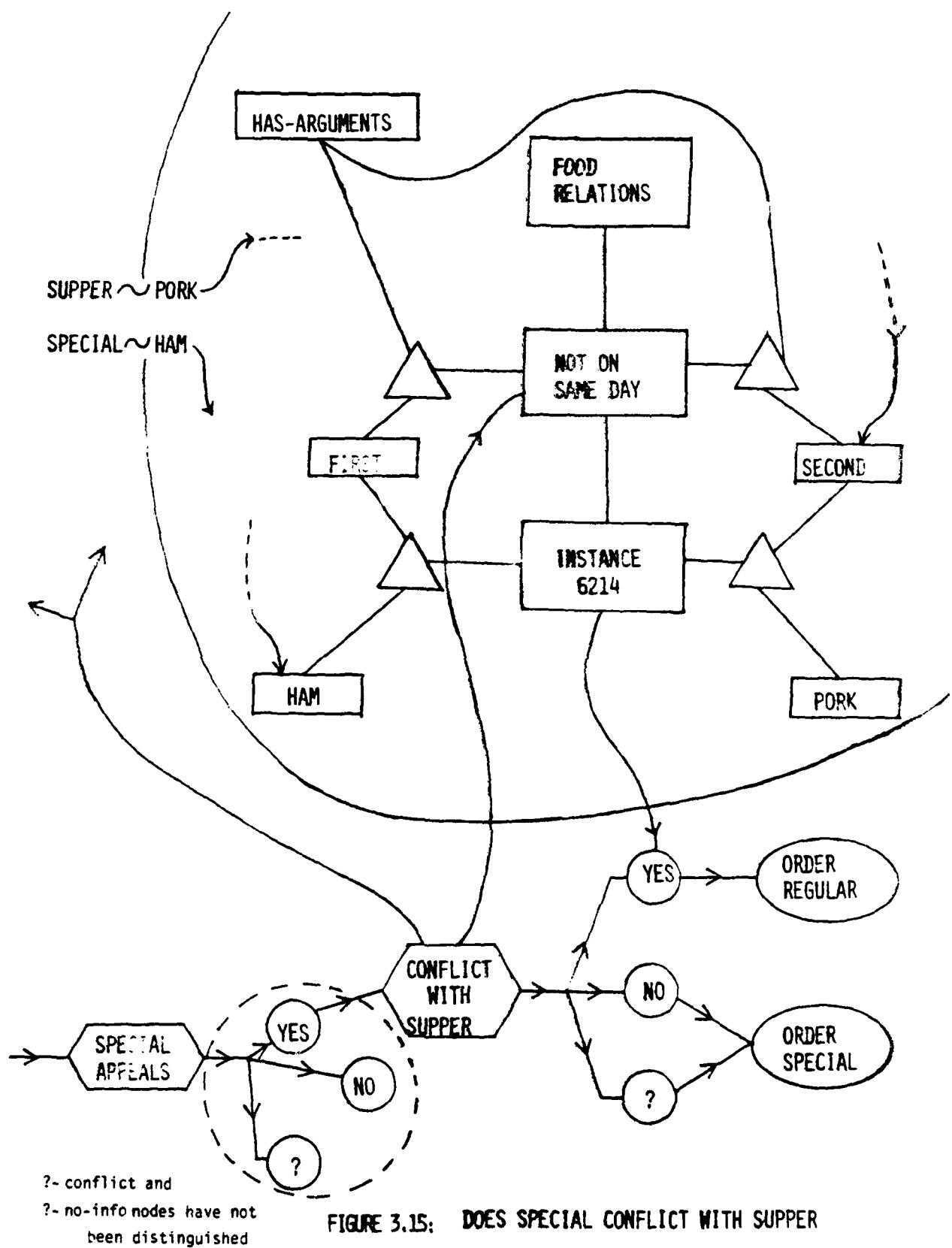
FIGURE 3.15: DOES SPECIAL CONFLICT WITH SUPPER

?- conflict and
?- no-info nodes have not
been distinguished
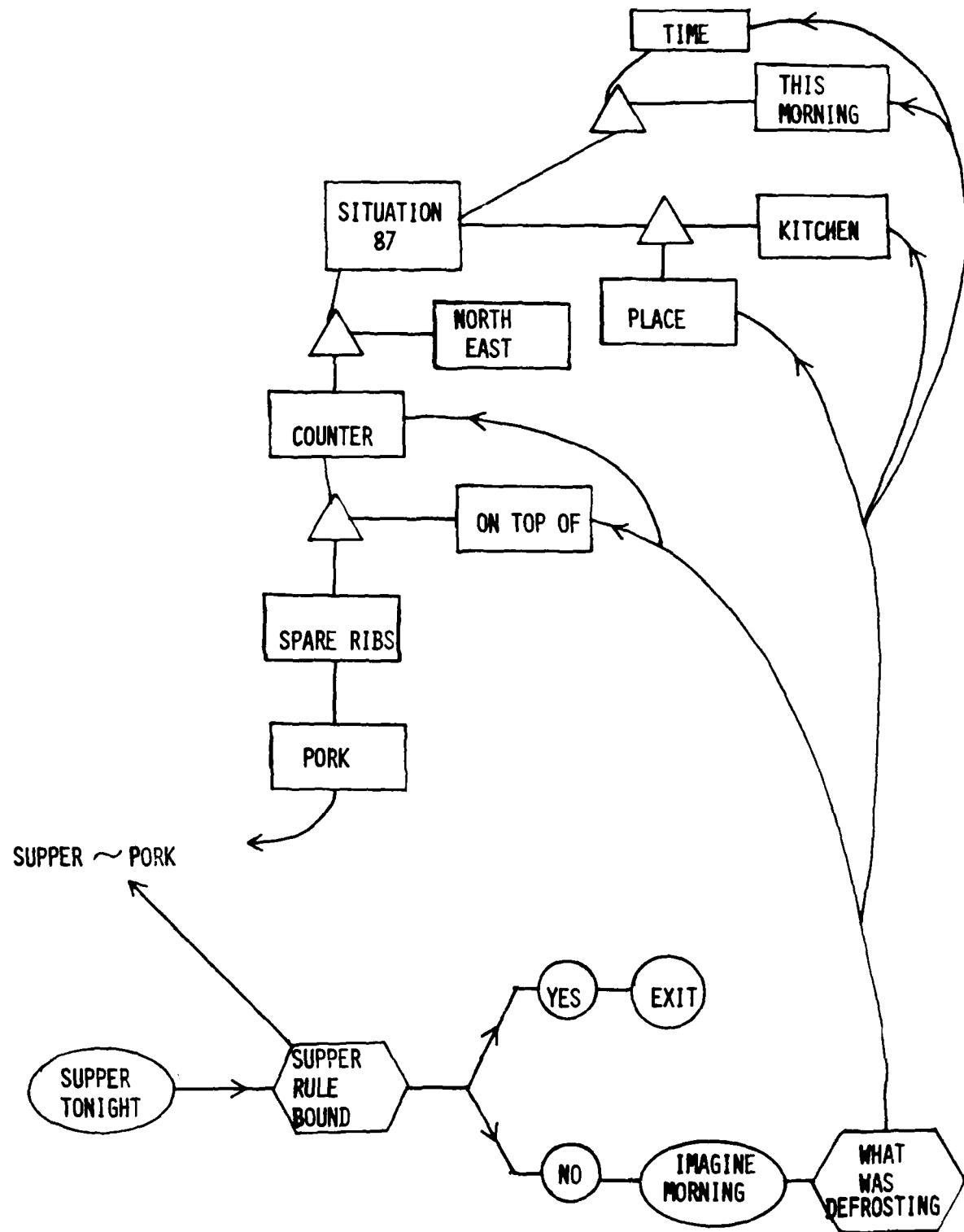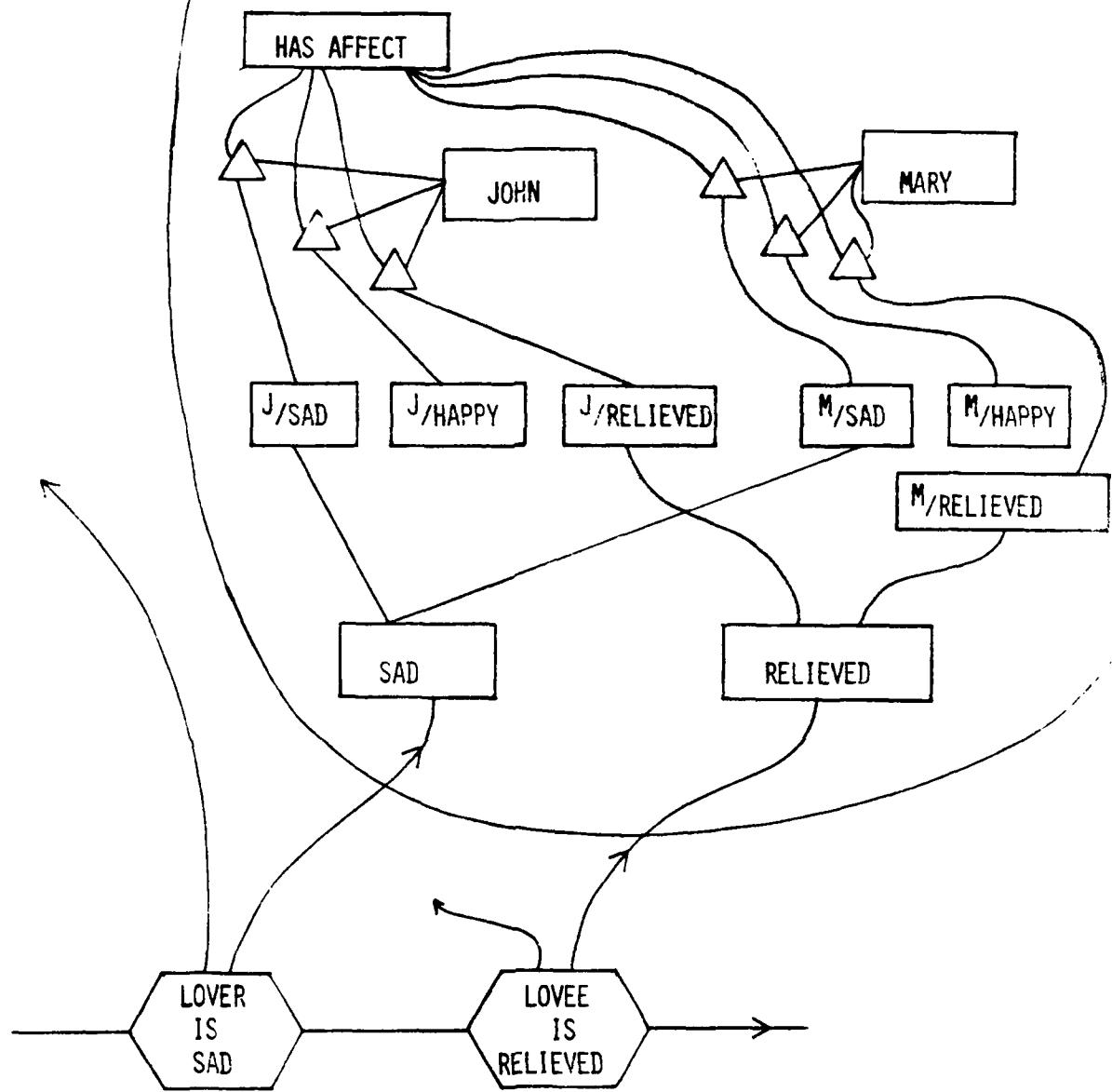
FIGURE 3.16

THE PARTING OF JOHN AND MARY

FIGURE 3.17

4. Implementation details

This section specifies a connectionist implementation of the knowledge representation and retrieval framework developed in the previous sections. It describes the computational characteristics of the units in the Memory Network and routines. General purpose routine networks have not been implemented but the current implementation can handle all the examples described in Section 3.3. The problem of establishing dynamic connections for role networks is dealt with in [Feldman 82a].

We have already described the logical connection patterns in our networks in Section 3 and will now specify the actual implementation and the dynamic behavior of the networks, i.e. the rules for the spreading and accumulation of activation. Spreading activation has been used as a parallel mechanism for propagating associative relevance over semantic networks [Quillian 68; Anderson 83]. In this work we have gone beyond this "facilitating" view of spreading activation and developed mechanisms for using it in a more controlled and structured manner to carry out limited inferences in active semantic networks.

As was described in Section 1.1, each active element (unit) in the connectionist framework is characterized by a potential function, a state function and an output function, each of whose values depend on the current value of the inputs and the previous values of the potential and the state. The potential function describes the dynamics of a unit's potential. The potential is a measure of a unit's activation and roughly corresponds to an integration of the input received by the unit in the recent past. The state function governs the transitions in a unit's state as a function of its current state, potential and inputs. The state of a unit controls the way it accumulates potential and sends output to other units. Finally, the output function describes the activation propagated by the unit on the basis of its state and potential.

A unit might receive input from a large number of other units. All inputs to a unit are not treated uniformly. Each unit has a number of input *sites* on which incident links impinge. Each site has an associated site function which maps its inputs into a single value. The potential, state and output functions act on the input via the values generated by the site functions.

A specification of site functions together with the potential, state and output functions specifies a unit type (not to be confused with the use of Type/Token in the representation scheme). Unit types are computationally distinct. Our connectionist implementation uses five unit types in the Memory Network and six unit types in the current implementation of routines. These roughly correspond to the different shapes of nodes used in the pictorial representation of networks in the previous sections. In what follows we will describe the various unit types in terms of their site, potential, state and output functions together with the weights used on links incident on different sites.

Before we proceed to describe the different unit types we specify the general form of the potential function used in these networks. The function common to most of the unit types is described below.

4.1 Potential Function

Each unit in the Memory Network and the Answer Network uses the following potential function:

$$p(t+1) = \nabla \times p(t) + i(t) \times [1.0 - \nabla \times p(t)] \qquad \textbf{eq·I}$$

Here $p(t)$ and $p(t+1)$ are the potentials at time t and $t+1$ respectively, $i(t)$ is the input to the unit at time t and ∇ is the decay constant. It is assumed that the input values and the parameter ∇ range between 0.0 and 1.0. The potential function returns a value bounded by 0.0 and 1.0 and may be interpreted as an *approximation* of a "leaky capacitor" where the potential decays with time and the integration is "damped" to saturate to a value of 1.0. A low value of ∇ means higher decay rates and consequently faster dynamics but lower steady state values. This function is a modification of the one reported in [McClelland & Rumelhart 81] and it has the desirable property that it does not lead to an oscillatory behavior when the potential reaches its maximum value of 1.0.

For any constant input I, the steady state value of the potential may be expressed as :

$$p_{ss} = I / [I \times \nabla - \nabla + 1] \qquad \textbf{eq·II}$$

Figure 4.1 is a plot of the steady state values of the potential for different input values with ∇ set to 0.6. Figure 4.2 plots the time response of the potential function for various values of inputs held constant in time. The figure also marks the time taken to reach 90% and 99% percent of the steady state values. In the design reported in this document ∇ was fixed at 0.6.

We now describe the different unit types starting with the simplest of these.

4.2 Characteristics of Query Units

The Query units have two sites - **enable** and **done**, and two states - **active** and **inert**. The site **enable** receives inputs from other units in routines while the site **done** receives inputs from the Answer Network. A Query unit is initially in the **inert** state but switches to the **active** state on receiving activation at the site **enable**. While in this state its potential remains fixed at 1.0 and its output equals its potential. It switches to the **inert** state when the input at the site **done** exceeds a certain threshold (currently 0.8). The detailed computational behavior of these units is specified in Figure 4.3.

4.3 Unit types in the Memory Network

The Memory Network is encoded using five distinct unit types. These are:

a) Enable Units: these units enable sites on binder units.

b) Relay Units: these are used to encode structural links.

c) Binder Units: these act as binders between properties, values and the owners (triangular nodes).

d) U-Binder Units: these encode the negative evidence and the uncertainty in the information about property values of a Type or Token.

e) Concept Units: these represent Types as well as Token nodes.

4.3.1 Enable Units

These units have an extremely simple behavior. They have only one site labelled **query**, which receives input from Query units in the routines, and two states: **active** and **inert**. These units are initially in the **inert** state and on receiving input from a Query unit switch to the **active** state. While in this state their potential is 1.0. The units revert to the **inert** state once they cease to get input. This behavior is summarized in Figure 4.4.

4.3.2 Relay Units

Relay units are used to encode structural links. As explained in Section 2.3.1, the role of structural links is to provide channels along which activation may spread during the retrieval process. Relay units provide the mechanism for encoding these channels. Figure 4.5a shows some units interconnected via three kinds of structural links and Figure 4.5b illustrates the actual encoding using Relay units. In general, every Concept unit has a number of Relay units associated with it; one for each *kind* of structural link relating it to other concept units. In the example shown in Figure 4.5b the units A. C and F own two Relay units each because they have two kinds of structural links associated with them. On the other hand, even though E has two outgoing links it owns only one Relay unit because both the links are of the same kind.

Relay units have three sites: **owner, upstream** and **enable**. The input from the owner concept is incident on the site **owner** while the inputs from other Relay units are incident on the site **upstream**. The activation propagated by the Relay unit is modulated by an enable input; if this input is off, the propagation is weak while if it is on, the propagation is strong. Currently, the enable signal has two levels but it is trivial to extend this to allow a range of modulation.

The details of the computational characteristics of these units are described in Figure 4.6. The design involves specifying three parameters:

α : the degree of attenuation per structural link and

σ_h and σ_l : the strength of activation corresponding to the high and low values of the enable signal.

The choice of parameters depends on the manner in which we want the activation to spread along structural links. In order to illustrate how the parameters are fixed, we will describe this process for the *is-instantiated-by* link. An *is-instantiated-by* link is the inverse of an *is-an-instance-of* link and is an example of a "topdown" link. Such

links normally have a weak "priming" or "facilitation" effect but pl., a more important role in the processing of Class II Queries (cf. Section 3.1).

We begin by fixing the weight of all incoming links to be 1.0. Next we make three design decisions in order to constrain the choice of parameters:

1) If a concept unit is at a potential of 1.0, then all its neighboring concept units linked via *is-instantiated-by* links should reach a potential of around 0.15, provided these links are enabled.

2) The effect should gradually decline as we move along successive links and after five levels should be around 0.05.

3) If the structural links are disabled, the potential of the immediate neighbor should remain in the vicinity of 0.05.

The value 0.15 in the first constraint is a measure of the evidence provided by a Type to each of its instances during the processing of a query that specifies a Type and seeks an instance of that Type (Class II Query). The low value of 0.05 in the third constraint is in keeping with the view that it is intended to model "priming" effects which are typically very weak. The exact values are not very crucial and an alternate set of comparable numbers could have been chosen.

These constraints may be expressed using the following equation:

$$I = \sigma \times \alpha^n \qquad \textbf{eq-III}$$

where I is the input available n levels away from a "source" (Concept unit) at potential 1.0. Using **eq-I** we find that the inputs required by a Concept unit to reach potentials of 0.05 and 0.15 are 0.02 and 0.07 respectively. Substituting these values in **eq-III** the constraints may be expressed as :

$$\sigma_h \times \alpha \simeq 0.07,$$
$$\sigma_h \times \alpha^5 \simeq 0.02 \text{ and}$$
$$\sigma_l \times \alpha \leq 0.02$$

There are many solutions satisfying the three constraints and the set of values chosen is: $\sigma_l = 0.03$, $\sigma_h = 0.08$ and $\alpha = 0.80$.

With this choice of values the resulting sequence of potentials at successive levels starting from a unit with potential 1.0 is:

1.0 0.15 0.12 0.09 0.07 0.06 -- with *is-instantiated-by* links enabled, and

1.0 0.05 0.04 0.03 0.02 0.02 -- with the links disabled.

This example suggests that the space of design parameters is underconstrained and to a great extent the choice of parameter values depends on the designer's judgement. However, it should also be obvious that there are precise rules constraining the set of possible choices. The approach has been to make a few arbitrary choices and then utilize these as constraints to obtain the remaining values.

Just as *is-instantiated-by* links may be characterized as "topdown" links, some other kinds of structural links may be classified as "bottomup". These links differ from "topdown" links in that when enabled they transmit much stronger levels of activation. A good example of a "bottomup" link is the *is-an-instance-of* link. The values of σ_l for "bottomup" links is the same as that for "topdown" links but the value of σ_h is significantly higher (0.71). Consequently, the resulting sequence of potentials at successive levels starting from a unit with potential 1.0 and with the *is-an-instance-of* links enabled is: 1.0 0.77 0.68 0.59 0.52 ... .

4.3.3 Binder Units

A Binder unit has three input sites labeled **p**, **o** and **v** (for property, owner and value respectively). Each site has two links incident on it. One link is an enable signal from the appropriate enable unit and the other is the input from a Concept unit. For instance, the site **p** receives a link from P-ENABLE and another from the Concept unit representing the property associated with the Binder. The enable signals are either ON or OFF and for a site to be active the associated signal should be ON. The computational characteristics of Binder units are described in details in Figure 4.7.

All the sites have the same site function. This function has the effect of raising the weights of the inputs when two or more sites are active. The potential function is as described in Section 4.1 with the input (i(t)) being the sum of the values returned by the three sites. Binder units have three states: **latent, hyper** and **refractory**. The unit enters the **hyper** state if two or more sites are active. In this state its potential grows at a high rate. The unit remains in the **hyper** state for 25 time steps after which it switches to the **refractory** state. In this state it ignores its inputs and its potential gradually decays. The unit switches back to the **latent** state once the potential falls below 0.05. The choice of 25 steps is based on the time it takes for a typical query to be processed from start to finish. The use of the **refractory** state is to bring back the system to a quiescent state. The unit transmits activation in all except the **latent** state and the value transmitted equals the unit's potential.

We outline the procedure for arriving at the weights and the input scaling parameter used in the **hyper** state.

A Binder unit serves two purposes. Its primary function is to detect simultaneous activation of two of its three neighbors and to activate the third when this happens. The secondary purpose of these units is to participate in "facilitation" effects by accumulating potential even in the absence of enable signals. The states **hyper** and **latent** characterize the two functions.

The weights of 0.05 and a scale factor of 6.0 are arrived at by considering the following constraints:

a) Even if all three units connected to the Binder unit are at a potential of 1.0 the potential of a Binder unit should remain well below 0.5 as long as the enable signals are off. (0.5 being the lower end of the range of potential at which a unit is considered to be "active" i.e. in a high state of activation).

b) If any two sites are active (enabled and getting more than half the maximum input), the potential of the Binder unit should reach at least 0.5 (a unit may be considered to be active above this value).

We fix the uppermost value of potential in the first constraint at 0.30. By using eq-II with P_{ss} set to 0.30, we find that the total input to the unit should be approximately 0.15. As there are three sites, this gives a weight of 0.05 per site. Notice that the choice of 0.30 for the maximum value of potential was arbitrary, one could as well have chosen 0.40 or 0.25 and obtained weights of 0.07 or 0.04 instead. As we said above the design is underconstrained and different sets of consistent weights may be chosen.

The scaling factor is calculated by considering the second constraint. i.e the unit should reach a value of at least 0.50 when two of its sites are active. Given that a site may be active even if the input is half its maximum value, we have the following condition:

If M_{avail} is the minimum value of input for which we want the unit to reach a potential of 0.5, then M_{avail} is given by: $2 \times 0.5 \times 0.05 = 0.05$

(two sites active each getting half of maximum input with weights of 0.05)

Using eq-II or Figure 4.1. it may be calculated that M_{req}, the total input needed for the unit to reach 0.5 is 0.30. This gives us a lower limit on the scaling factor of M_{req} / M_{avail} which is $0.30/0.05 = 6.0$.

4.3.4 U-Binder units

In Section 2.4.1 we described how negative evidence is encoded using special Binder nodes. These Binder nodes were activated only if an inapplicable property value was specified and on being activated they sent an inhibitory response to their owner (a Concept unit). The U-Binder units encode these special Binder nodes. The detailed description appears in Figure 4.8 and we will only describe the differences between the Binder and the U-Binder units.

Besides the three sites - **p**, **v** and **o**, U-Binder units have an additional site labeled **b**. This site receives inputs from Binder units related to the relevant property and owned by the same Concept unit that owns the U-Binder unit. The site **b** becomes active if the input indicates that a Binder is in the **hyper** state. Unlike the corresponding sites on Binder units, the sites **v** and **o** of U-binder units receive inputs only from the appropriate ENABLE units and are considered active if the enable signals are ON.

A U-Binder unit enters the **hyper** state if **b** is *not active* but both **p** and **v** are active (**b** is an inhibitory site). In this state the unit sends output after a delay of one time step.

4.3.5 Concept units

These units represent Types as well as Tokens in the Memory Network and their basic function is to integrate the incoming activation. The computational features of the Concept units are described in Figure 4.9. These units have a site for structural links (**relay**), three sites for receiving inputs from Binders (**bv bp and bo**) and a site (**query**) for receiving inputs from Query units in the routines.

The links incident on the site **relay** have a weight of 1.0 while the weights on links at the site **bp** are 0.80. The links incident on sites **bo** and **bv** are evidential links. The weight associated with a positive evidential link of strength e is such that it will drive the potential of the target unit to e in the absence of other inputs. The values of positive evidential weights are obtained using **eq-I** and they may range from 0.0 to 1.0. The weights on links from U-Binder units are negative and their value is equal to the strength of negative evidence.

While the positive inputs are simply added, the negative inputs from U-Binders are treated differently by Concepts and their contribution is given by:

$$\log(\ \text{abs(input)}), \qquad \text{bounded between -2.0 and 0.0}$$

The rationale for using the logarithmic function is as follows: for a given property, if there is very little uncertainty that the values specified by the Binders are the only possible values and a wrong value is specified, then the negative evidence for the Concept should be high. However, if the uncertainty is high then the negative evidence should be low. In the limiting cases the negative evidence should be 0 (total uncertainty) or $-\infty$ (no uncertainty). A negative evidence of $-\infty$ is an idealization and we use a bounded but high negative value.

Unlike the Binder unit, the sites on the Concept units have multiple inputs incident on them. Different sites have different site functions and information about potential function, state function and the output function is described in Figure 4.9.

4.4 Characteristics of the units in the Answer Network

Having specified the connectivity and also the computational characteristics of units in the Memory Network, we next examine the units in the Answer Network.

There are five kinds of units in this network. These are the Response units, the Max-calculator units, the Conflict-detector units, the ?-conflict units and the ?-no-info units.

4.4.1 Max-calculator units

A Max-calculator unit has one site namely, **max**. All inputs from Response units are incident on this site. The site-function returns the maximum of these inputs and the unit's potential takes on the value returned by the site-function. The output of the unit is equal to the value of its potential. Refer to Figure 4.10 for details.

4.4.2 Conflict-detector units

A Conflict-detector unit has two sites: **sum** and **enable**. The site **enable** receives input from the Query unit that initiates the query and it simply returns the input value while the site **sum** receives inputs from all the Response units and returns the sum of these values. Initially, a Conflict-detector unit starts out in the inert state and ignores all inputs at the site **sum**. On receiving a positive input at **enable** it switches to the **initial** state. In this state the site **sum** processes its inputs but the unit maintains a potential of 1.0. If the value returned by **sum** exceed 1.0 the unit switches to the **interim** state and its potential equals the inverse of the value returned by **sum** (subject to a maximum of 1.1 and minimum of 0.1). The unit switches to the **inert** state whenever the input at **enable** falls to 0.0. The output of this unit is equal to its potential. Figure 4.11 specifies this information in detail.

4.4.3 Response units

The Response units have four sites. One for inputs from the Memory Network (**m-net**), one for input from the Conflict-detector unit (**sum**), one for input from the Max-calculator unit (**max**) and the last one (**enable**) for inputs from Query units. All the sites except **enable** simply return the value received by them. The site **enable** returns the maximum value received by it. The unit starts out in the inert state. In this state it ignores all inputs at sites **m-net**, **max** and **sum**. The unit switches to the **active** state if the site **enable** returns a positive value. In this state the unit maintains a potential but the growth of potential is governed by the values returned by the different sites as described below:

1. The Input is defined as the current potential minus the input from the Max-calculator unit plus the input from the unit in the Memory Network.

2. The value of ∇ is set to the input received from the Conflict-detector unit.

Once the potential exceeds a preset value (currently 0.80), the unit switches to the **winner** state. If the input at **enable** falls to 0.0 the unit reverts to the inert state and its potential gradually decays to 0.0. The output of these units equals its potential. Figure 4.12 summarizes this information.

The network composed of Response units, the Conflict-detector unit and the Max-finder unit is an efficient implementation of a winner-take-all network (WTA) where the competition is set up between the Response units. This design of a WTA has many advantages over earlier ones. First, it uses only $O(n)$ links as against $O(n^2)$ for a WTA of size n. Second, it has a built in preference for finding a single winner. If more than one Response unit is getting strong evidence, the variable parameter ∇ has a dampening effect on the growth of the potential of the response units and none of the units reaches a very high value of potential. Thus, if for some reason a clear winner is not immediately available the competition is automatically extended. This permits evidence to accumulate over varying periods of time. This feature is very important if evidence from multiple levels in the Memory Network is to be integrated. The computations in the Memory Network may now perform a hierarchical decision making process (but parallel within a level in the hierarchy).

However, there are cases in which there is simply no clear winner and none of the Response units ought to dominate. Two such cases identified in Section 3.1

were: a) none of the Response units get evidence or b) more than one Response units get nearly equal evidence. The following unit types handle these two cases.

4.4.4 ?-Conflict Units

?-Conflict units are designed to win if more than one Response unit is receiving evidence from the Memory Network and yet no single Response unit is able to dominate the competition. These units have two sites **sum** and **enable**. The former receives input from the Conflict-detector unit and returns the inverse of the value and the latter returns the maximum value of its inputs. The potential of these units grows if the value returned by **sum** exceeds 1.0 and the site **enable** is receiving positive input. Otherwise the potential gradually decays to 0.0. The detailed description is given in Figure 4.13.

4.4.5 ?-No-info Units

A ?-no-info unit gains potential if none of the Response units receive any evidence from the Memory Network. The unit's behavior is very simple and is described in Figure 4.14. A positive input at site **max** indicates that at least one Response unit has a positive potential. In this case, the unit's potential decays from its current value. If the input at site **max** is 0.0 then the unit gains potential. The only significant point is that there are two rates at which a ?-no-info unit may accumulate potential. If only local evidence in the Memory Network is to be used during the query, the rate of accumulation may be increased by activating the site **accum-rate**.

4.5 Conclusion

The implementation described herein is evolving over time. However, the main features have been stable for nearly a year. Work is in progress to formally characterize the computations performed by the units in the Memory and Answer Networks and to relate the dynamics of these networks to a formal theory of evidence. A discussion of these issues appears in Section 5.
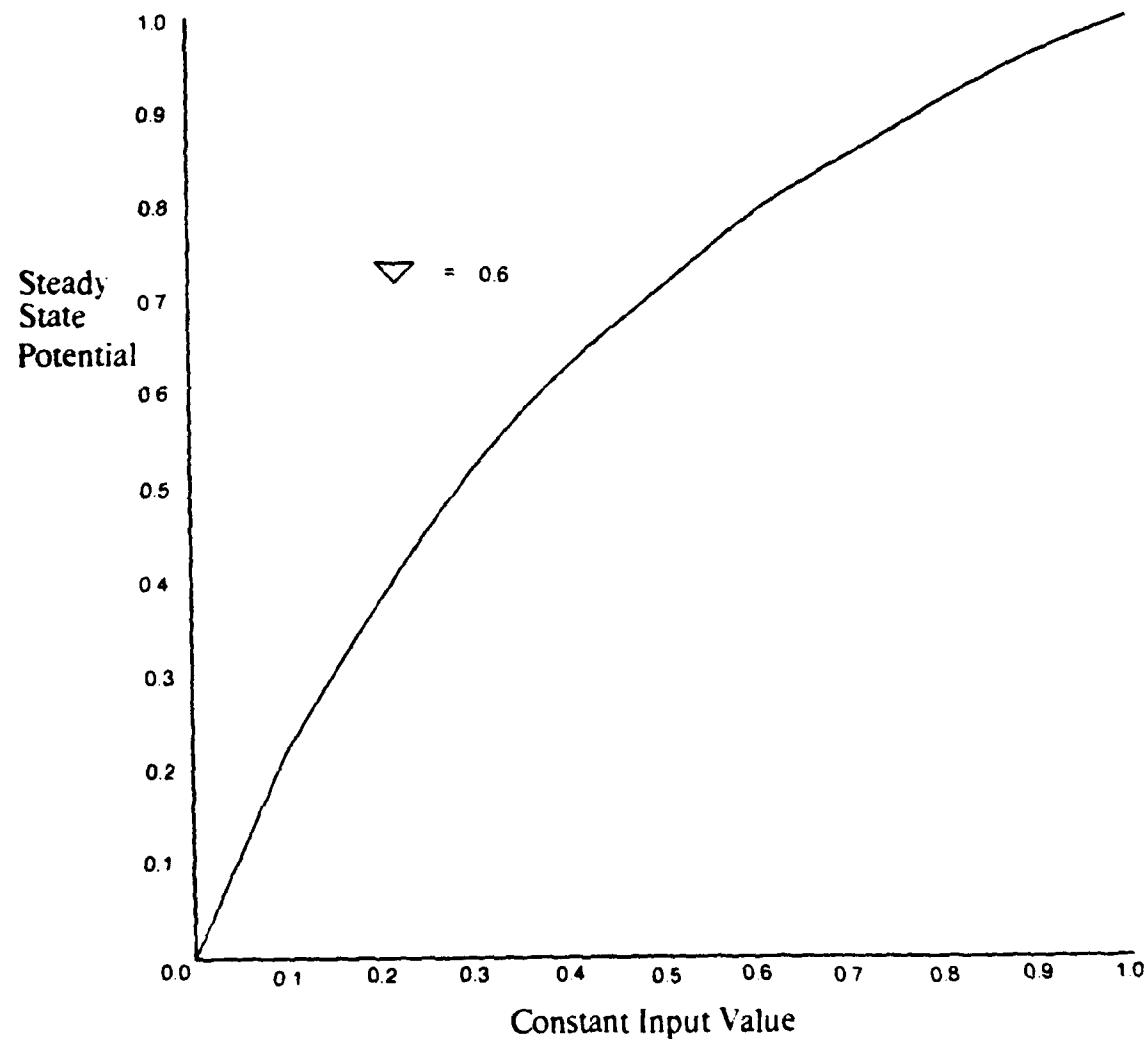
Figure 4.1: Steady State Potential for Various Input Values
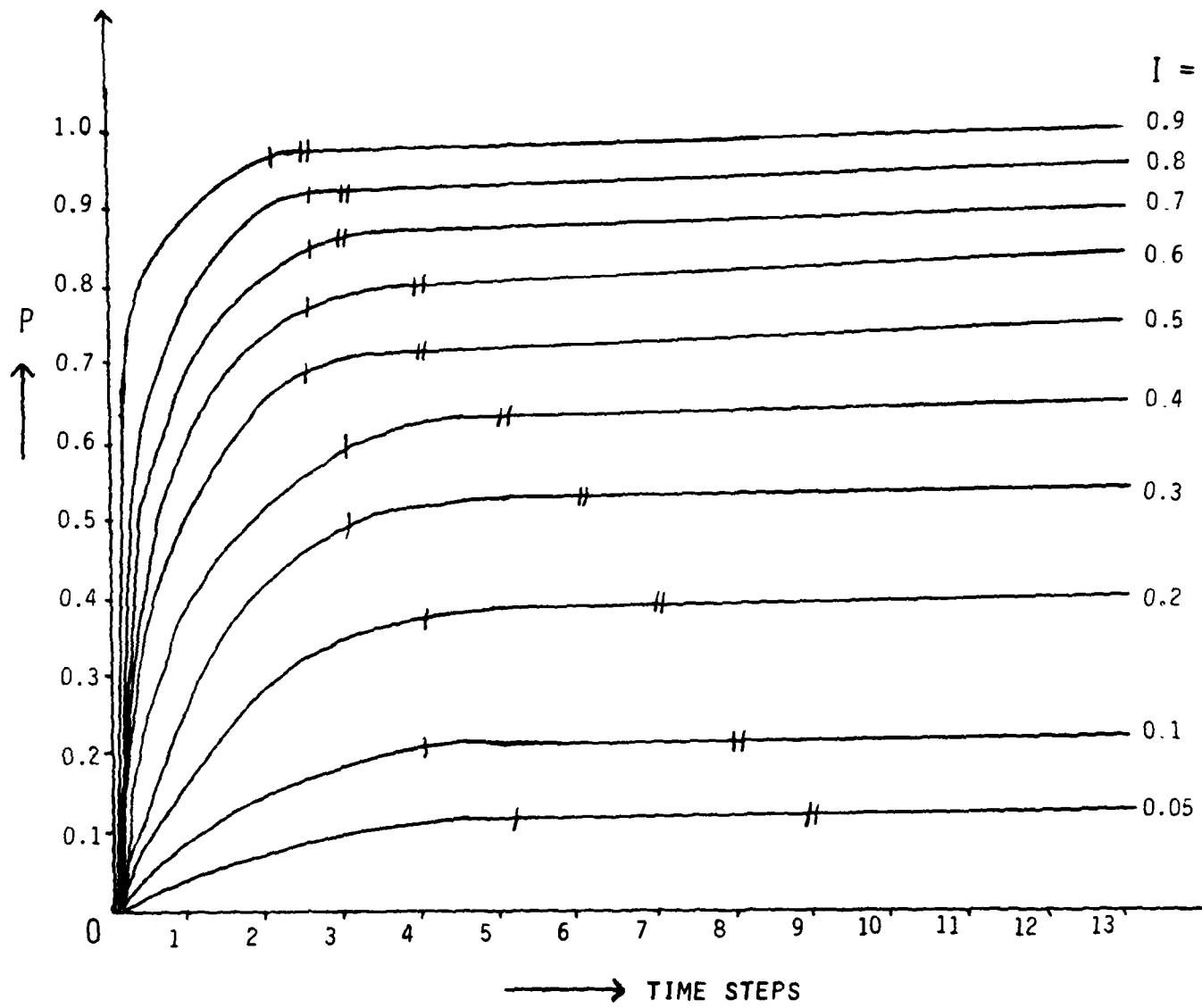
FIGURE 4.2 TRACE OF POTENTIAL FOR VARIOUS VALUES OF INPUTS
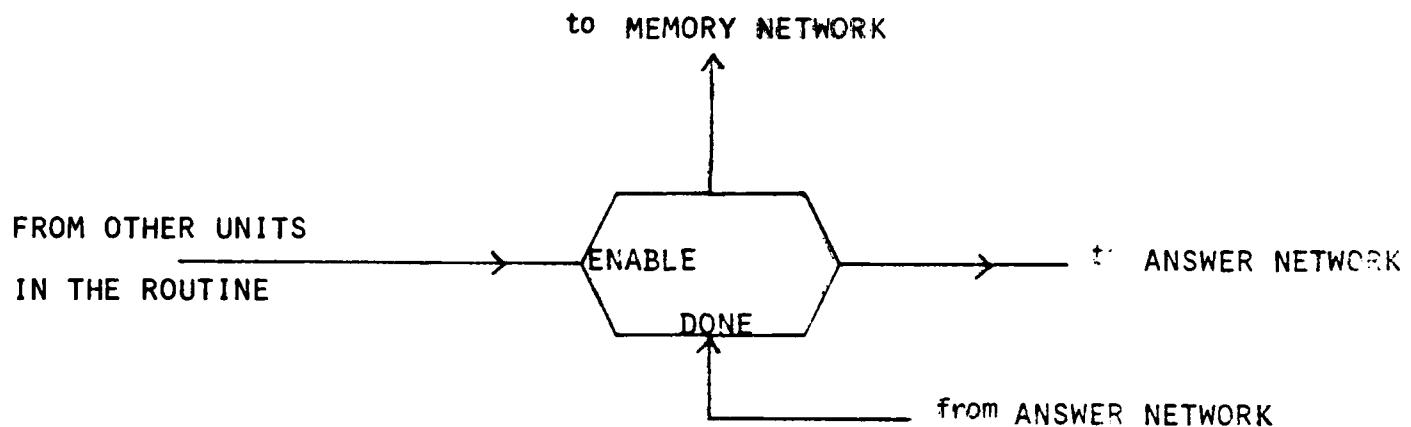
to MEMORY NETWORK

FROM OTHER UNITS

IN THE ROUTINE

ENABLE

DONE

to ANSWER NETWORK

from ANSWER NETWORK

WEIGHTS

All incoming links have a weight of 1.0

SITE FUNCTIONS

ENABLE: Returns the value of the highest input
DONE: Same as above

POTENTIAL FUNCTION

$p(t+1)$ $\leftarrow 1.0$ if $q(t) =$ **active**   ; $p(t+1)$ is the potential at time $t+1$
          $\leftarrow 0.0$ otherwise   ; $q(t)$ is the state at time t

STATE FUNCTION

ENABLE > 0

START
STATE

INERT

ACTIVE

DONE > 0.8*

*DESIGN PARAMETER
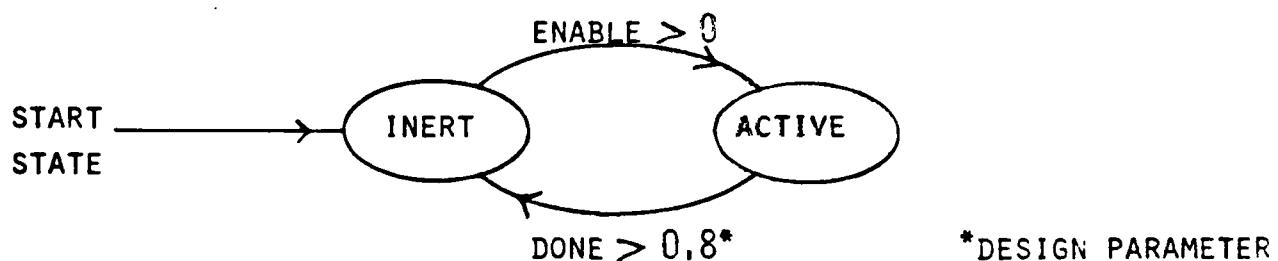
OUTPUT FUNCTION

$o(t) \leftarrow p(t)$

FIGURE 4.3: Query Unit

from QUERY UNITS   ⟶ QUERY   ⟶ to BINDER UNITS
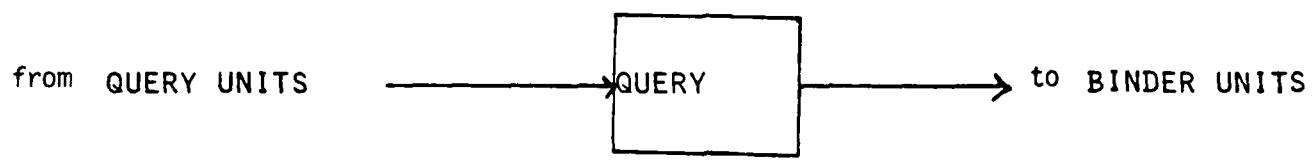
WEIGHTS

    All incoming links have a weight of 1.0

SITE FUNCTIONS

    QUERY: Returns the value of the highest input

POTENTIAL FUNCTION

$$p(t+1) \quad \leftarrow 1.0 \text{ if } q(t) = \textbf{active}$$

$$\leftarrow 0.0 \text{ otherwise}$$

STATE FUNCTION

START STATE ⟶ INERT   QUERY > 0   ACTIVE   QUERY = 0
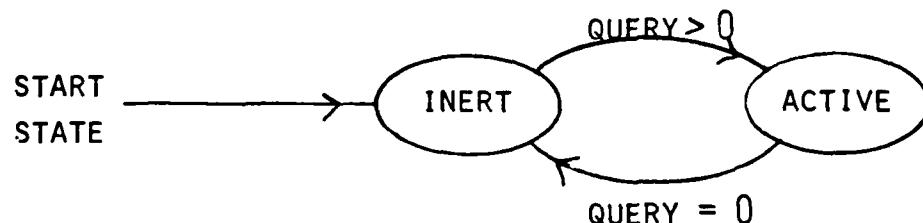
OUTPUT FUNCTION

$$o(t) \leftarrow p(t)$$

FIGURE 4.4: Enable Units
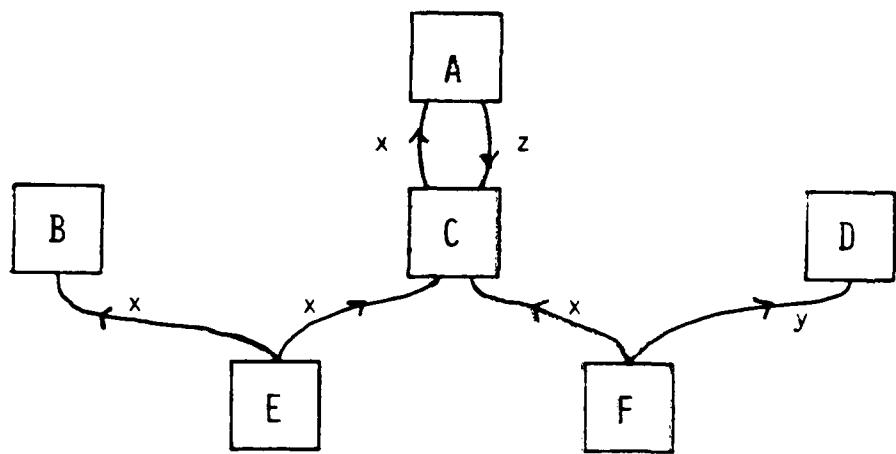
x, y AND z ARE DIFFERENT KINDS OF STRUCTURAL LINKS

FIGURE 4.5 A
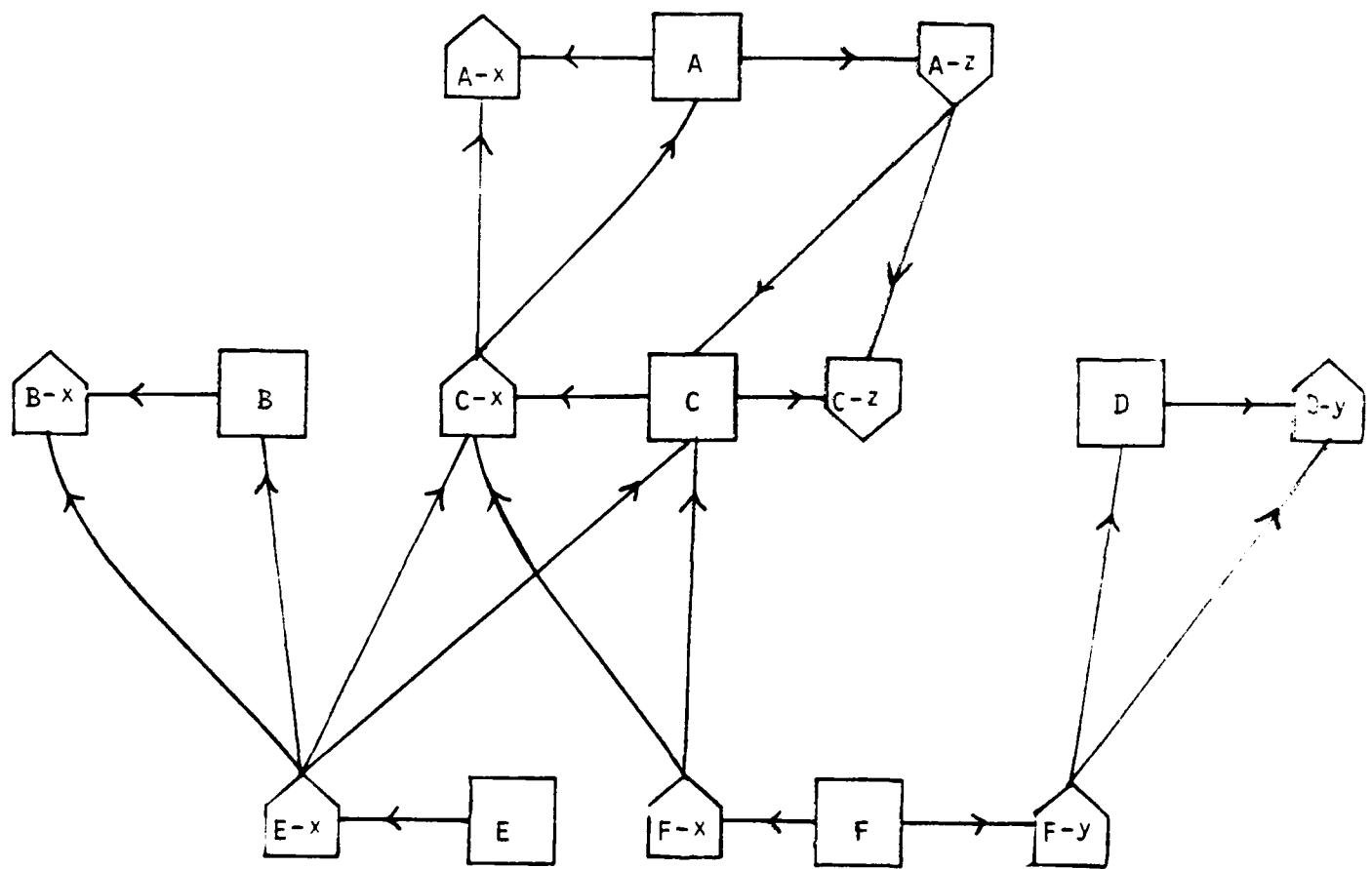
FIGURE 4.5 B
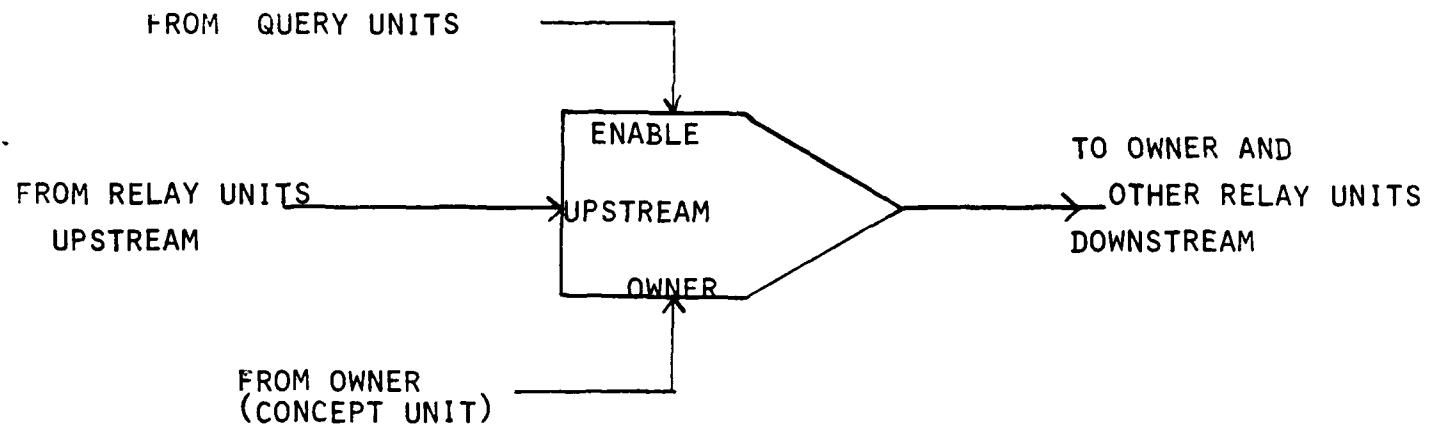
FROM QUERY UNITS

ENABLE

FROM RELAY UNITS
UPSTREAM

UPSTREAM

OWNER

TO OWNER AND
OTHER RELAY UNITS
DOWNSTREAM

FROM OWNER
(CONCEPT UNIT)

WEIGHTS

All incoming links have a weight of 1.0

SITE FUNCTIONS

ENABLE:	Returns the value of the highest input
UPSTREAM:	Returns the sum of all inputs
OWNER:	Returns input $\times \sigma_h$ if state = **enabled**
	else returns input $\times \sigma_l$

POTENTIAL FUNCTION

$p(t) \leftarrow \max (\text{UPSTREAM, OWNER})$

STATE FUNCTION

If ENABLE > 0.0 then $q(t)$ = **enabled**

else if potential > 0.0 then $q(t)$ = **active**

else $q(t)$ = **latent**

OUTPUT FUNCTION

$o(t) \leftarrow p(t) \times \alpha$

Currently $\alpha = 0.80$ and $\sigma_l = 0.03$
"topdown" links have $\sigma_h = 0.08$ while
"bottomup" links have $\sigma_h = 0.71$

FIGURE 4.6: Relay Units

P-ENABLE SIGNAL

FROM PROPERTY ⟶ P

O-ENABLE SIGNAL

FROM OWNER ⟶ O

V-ENABLE SIGNAL

FROM VALUE ⟶ V

TO PROPERTY, OWNER,

VALUE AND U-BINDER

UNITS
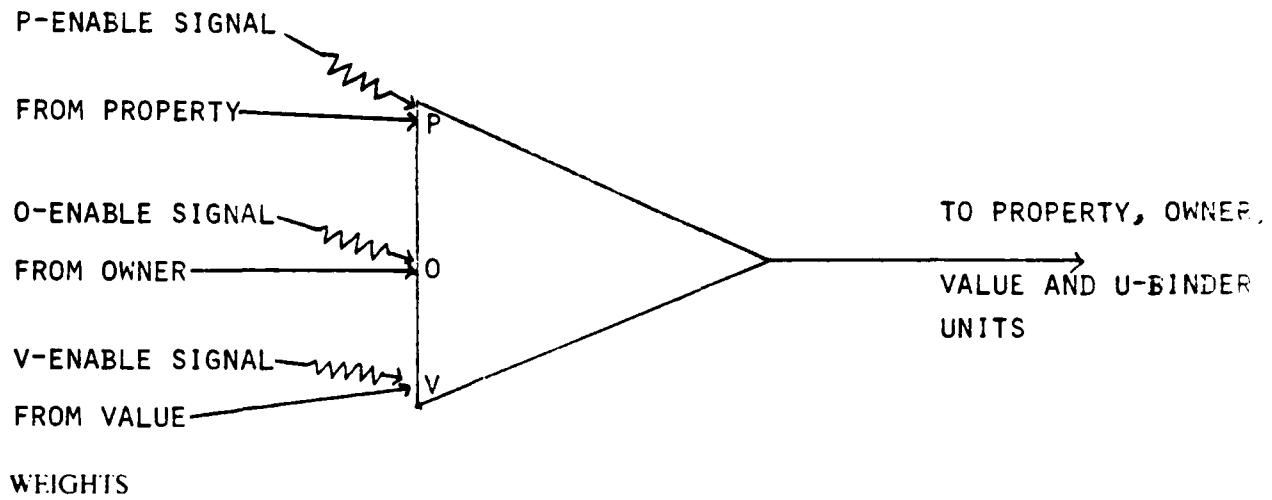
WEIGHTS

Enable signals are either ON or OFF, other links have a weight of 0.05

SITE FUNCTIONS

All sites have the same site function:
If state = **hyper** return input × 6.0
if state = **refractory** return 0.0
if state = **latent** return input

A site is active if the enable signal is on and the input is at least half its maximum value.

POTENTIAL FUNCTION

$$p(t+1) \leftarrow \nabla \times p(t) + i(t) \times (1.0 - \nabla \times p(t))$$

$\nabla = 0.6$, $i(t) = $ sum of values returned by all the sites.

STATE FUNCTION

2 OR MORE SITES ACTIVE

START STATE ⟶ LATENT → HYPER

IF POTENTIAL FALLS BELOW 0.05

REFRACTORY

AFTER 25** TIME STEPS
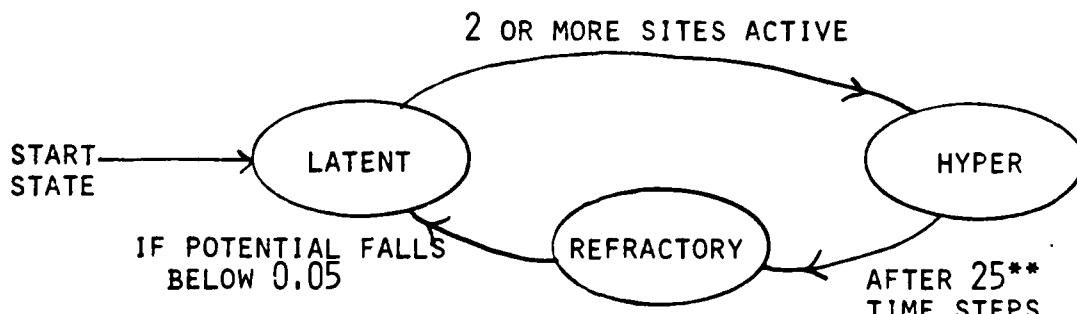
OUTPUT FUNCTION

If $q(t) = $ **latent** then $o(t) \leftarrow 0.0$
                       else $o(t) \leftarrow p(t)$

**DESIGN PARAMETER

FIGURE 4.7: Binder Unit

P-ENABLE SIGNAL

FROM PROPERTY — P

O-ENABLE SIGNAL —WWW→ O

B

V-ENABLE SIGNAL —WWW→ V

TO OWNER AND
VALUE UNITS
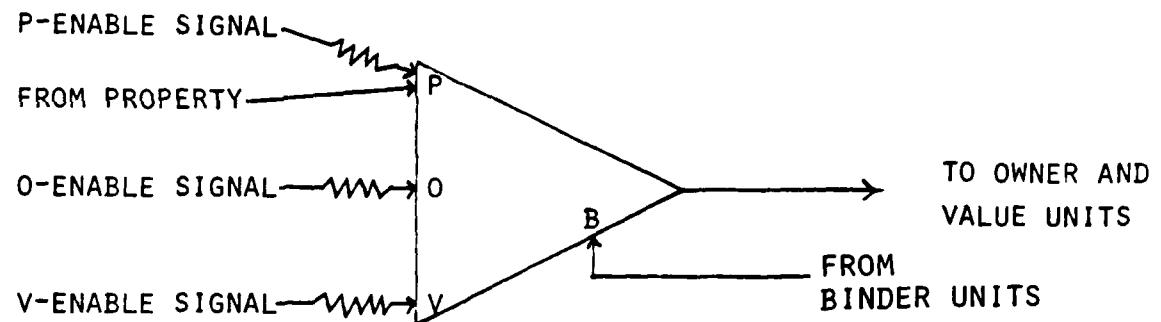
FROM
BINDER UNITS

WEIGHTS

Enable signals are either ON or OFF, link from Property have a weight of 0.05 while links from Binders have a weight of -1.0

SITE FUNCTIONS

All sites ignore all inputs in refractory state.

B:	If any input less than half the minimum value then site is active. Returns the minimum input value.
P:	If P-Enable ON and input greater than half the maximum then site active. Returns the input value if state = **latent** but returns 6 × input if state = **hyper**.
O,V:	Site active if Enable signal is ON.

POTENTIAL FUNCTION

$$p(t+1) \leftarrow \nabla \times p(t) + i(t) \times (1.0 - \nabla \times p(t))$$

$\nabla = 0.6$, $i(t)$ = sum of values returned by all the sites.

STATE FUNCTION

B NOT ACTIVE BUT
P AND V ACTIVE

START
STATE → LATENT

HYPER

REFRACTORY

IF POTENTIAL
FALLS BELOW 0.05

AFTER 25 TIME STEPS
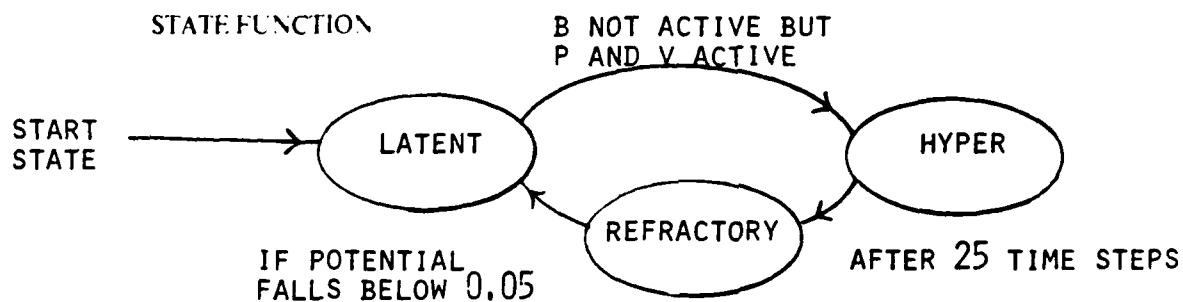
OUTPUT FUNCTION

If $q(t) =$ **latent**   then $o(t) \leftarrow 0.0$

else* $o(t) \leftarrow p(t)$

* No output in the time step immediately following the entry into the **hyper** state.

FIGURE 4.8: U-Binder Unit

FROM QUERY UNITS ──────────┐
                           ▼
                       ┌─────QUERY──────┐
FROM BINDER UNITS ⎧──→ BP   │
                   ⎨──→ BV   ├────────→ TO RELAY AND
                   ⎩──→ BO   │   BINDER UNITS
                       └─────RELAY──────┘
                              ▲
FROM RELAY UNITS ─────────────┘
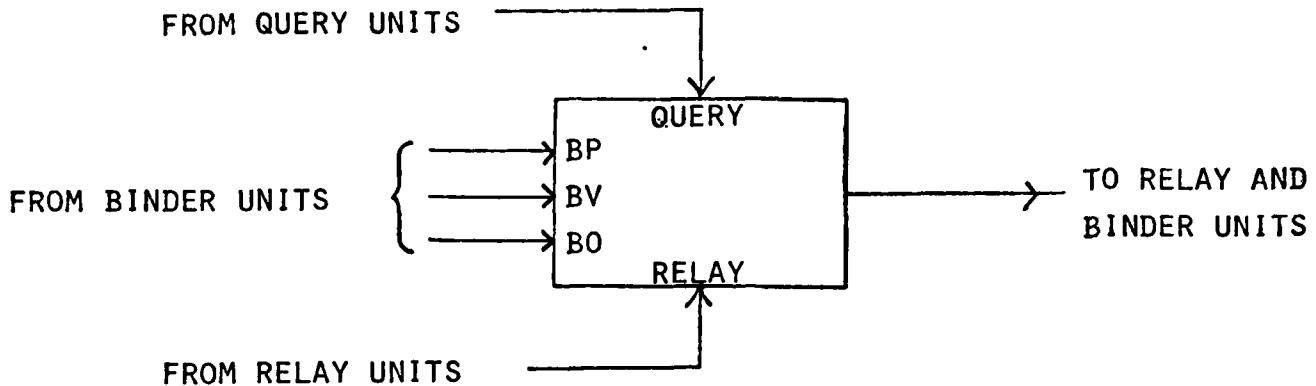
WEIGHTS

Links incident at site QUERY and RELAY have a weight 0f 1.0,
Links incident at site BP have a weight 0f 0.80.
Links at site BV and BO are evidential links and their weights vary from 0.0 to 1.0.
(See section 4.3.5 for an explanation).

SITE FUNCTIONS

QUERY: Returns input value
RELAY, BV: Returns sum of input values
BP: Returns the value of the highest input
BO: Add each positive value and add log(mod(input value)) for negative value *.

* The contribution of each negative value is bounded by -2.0.

POTENTIAL FUNCTION

$$p(t+1) \leftarrow \nabla \times p(t) + i(t) \times (1.0 - \nabla \times p(t))$$

$\nabla = 0.6$, $i(t) =$ sum of values returned by all the sites.

STATE FUNCTION

$q(t) =$ **broadcast** if $p(t) > 0.0$
            else state $=$ **latent**

OUTPUT FUNCTION

If $q(t) =$ **latent**   then $o(t) \leftarrow 0.0$
If $q(t) =$ **broadcast** then $o(t) \leftarrow p(t)$

FIGURE 4.9: Concept Unit

TO RESPONSE, ?-NO-INFO
AND QUERY UNITS

MAX

FROM RESPONSE UNITS
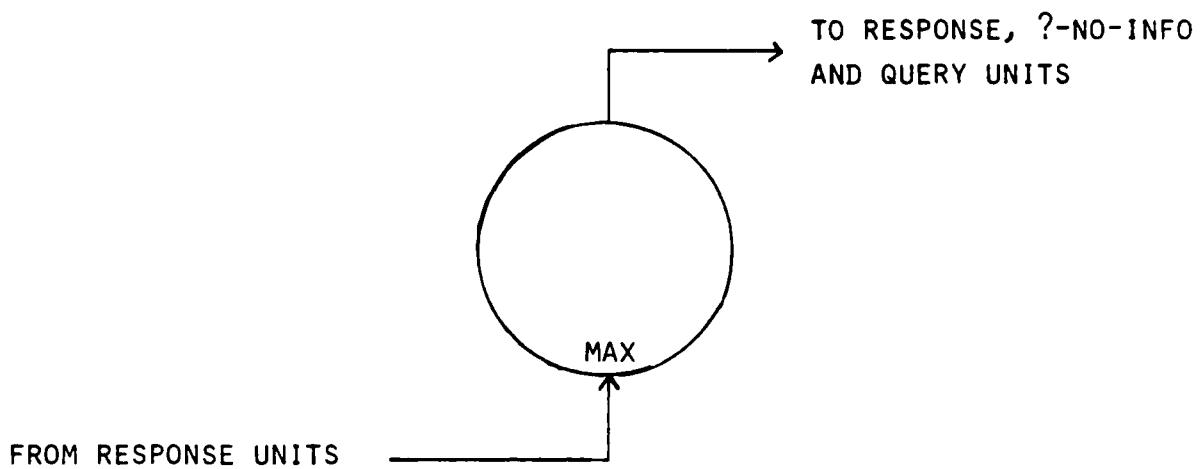
WEIGHTS

All incoming links have a weight of 1.0

SITE FUNCTIONS

MAX: Returns the value of the highest input

POTENTIAL FUNCTION

p(t)   ← MAX

STATE FUNCTION

Single State

OUTPUT FUNCTION

o(t) ← p(t)

FIGURE 4.10: Max-calculator Unit

FROM QUERY UNITS ⟶ ENABLE ⟶ TO RESPONSE AND
?-CONFLICT UNITS

SUM ↑

FROM CONFLICT-DETECTOR⟶
   UNIT

WEIGHTS

All incoming links have a weight of 1.0
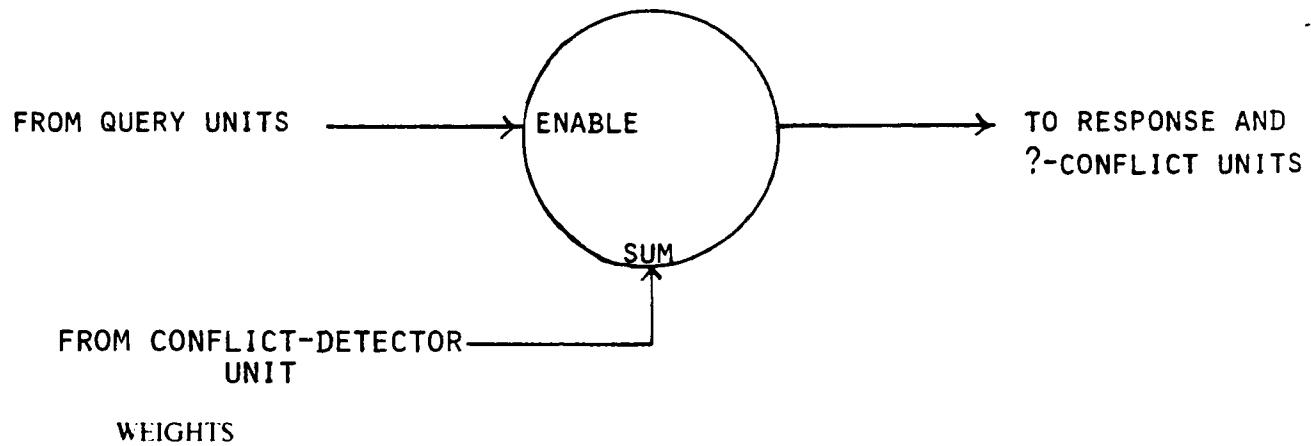
SITE FUNCTIONS

ENABLE: Returns the value of the highest input
SUM: Returns the sum of all the input values

POTENTIAL FUNCTION

$p(t+1)$ ← ∇×p(t) if q(t) = **inert**
        ← 1.0 if if q(t) = **initial**
        ← max (0.1, (min 1.1, 1/sum)) if q(t) = **interim**

STATE FUNCTION
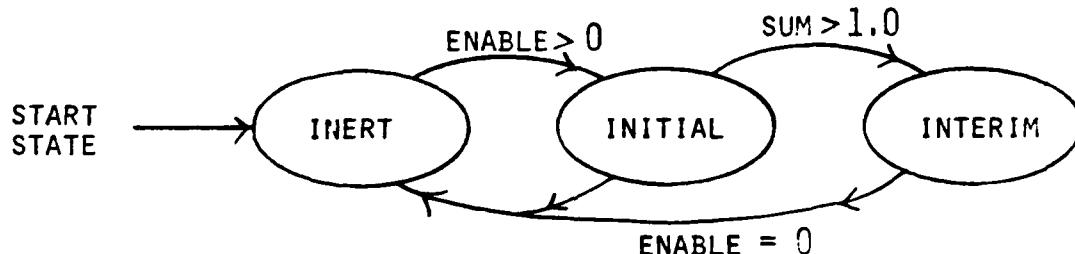
OUTPUT FUNCTION

o(t) ← p(t)

FIGURE 4.11: Conflict-detector Unit
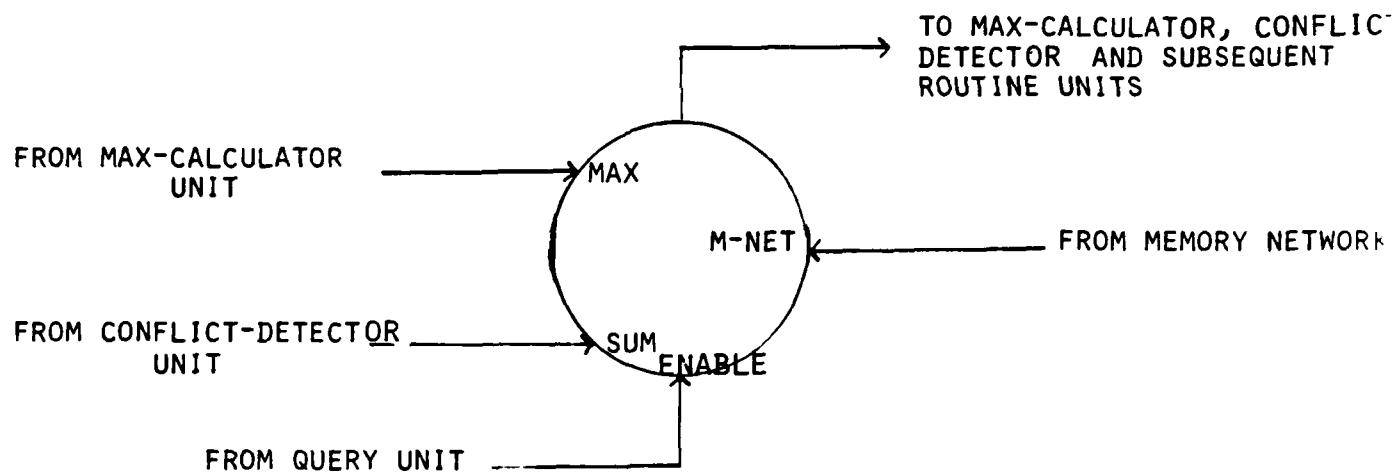
TO MAX-CALCULATOR, CONFLIC'
DETECTOR AND SUBSEQUENT
ROUTINE UNITS

FROM MAX-CALCULATOR
UNIT

MAX

M-NET ← FROM MEMORY NETWORk

FROM CONFLICT-DETECTOR
UNIT

SUM
ENABLE

FROM QUERY UNIT

WEIGHTS

All incoming links have a weight of 1.0

SITE FUNCTIONS

ENABLE: Returns the value of the highest input

SUM, MAX and M-NET: Return the input values

POTENTIAL FUNCTION

$p(t+1)$ ← $\delta \times p(t)$ if $q(t) = $ **inert**

← $\nabla(t) \times p(t) + \alpha \times (M\text{-}NET + p(t) - MAX) \times (1.0 - \nabla(t) \times p(t))$

otherwise.

$\nabla(t) = SUM, \alpha = 0.35$ and $\delta = 0.6$

STATE FUNCTION

If ENABLE > 0.0   then   if $p(t) > 0.80$ then $q(t) = $ **winner**

else $q(t) = $ **active**

else $q(t) = $ **inert**

OUTPUT FUNCTION

$o(t) \leftarrow p(t)$

FIGURE 4.12: Response Unit

FROM CONFLICT-DETECTOR
UNIT   → SUM

TO PRECEDINE QUERY
UNIT AND ANY
SUBSEQUENT ROUTINE
UNITS

ENABLE

FROM QUERY UNITS ————

WEIGHTS

All incoming links have a weight of 1.0
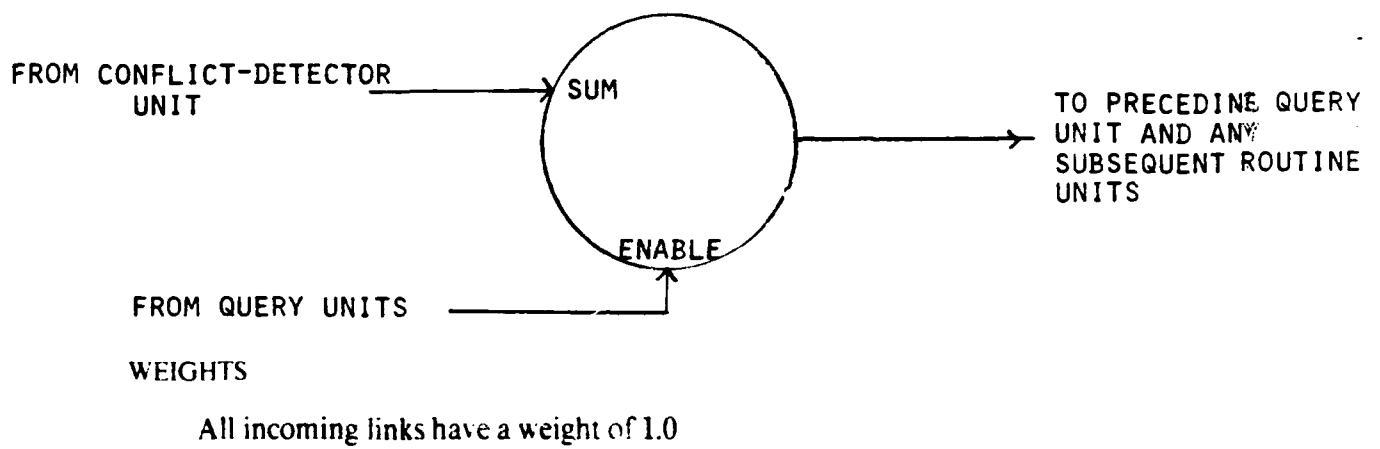
SITE FUNCTIONS

ENABLE: Returns the highest input value
SUM: Returns   (1.0/input value)

POTENTIAL FUNCTION

if $q(t)$ = **inert** then
    $p(t+1) \leftarrow \nabla \times p(t)$

If $q(t) \neq$ **inert** then
    $p(t+1) \leftarrow p(t) + (SUM - 1.0)$ if $SUM > 1.0$
 $\leftarrow \nabla \times p(t)$ if $SUM \leq 1.0$: $\nabla = 0.6$

STATE FUNCTION

If ENABLE > 0.0 then   if $p(t) > 0.80$ then $q(t)$ = **winner**
                                                else $q(t)$ = **latent**
                  else $q(t)$ = **inert**

OUTPUT FUNCTION

$o(t) \leftarrow p(t)$

FIGURE 4.13: ?-Conflict Units

FROM MAX-CALCULATOR UNIT ──────→ MAX

FROM QUERY UNITS ──────→ ACCUM-RATE

                              ENABLE

FROM QUERY UNITS ──────┘

TO PRECEDING QUERY
UNIT AND ANY SUBSEQUEN
ROUTINE UNITS
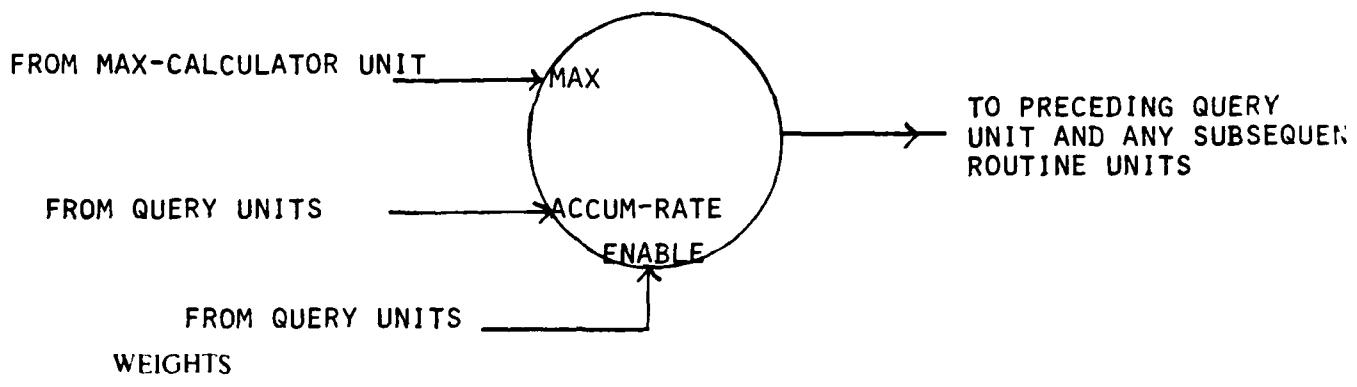
WEIGHTS

All incoming links have a weight of 1.0

SITE FUNCTIONS

ENABLE. MAX, and ACCUM-RATE: All return the highest input value

POTENTIAL FUNCTION

if $q(t) = $ **inert** then
   $p(t+1) \leftarrow \nabla \times p(t)$

If $q(t) \neq $ **inert** then
   $p(t+1) \leftarrow p(t) + \beta$ if MAX $= 0.0$
           $\leftarrow \nabla \times p(t)$ if MAX ≥ 0.0

$\nabla = 0.6$:   if ACCUM-RATE > 0.0 then $\beta = 0.08$ otherwise β equals 0.03

STATE FUNCTION

If ENABLE > 0.0 then   if $p(t) > 0.80$ then $q(t) = $ **winner**
                                                                else $q(t) = $ **latent**
                        else $q(t) = $ **inert**

OUTPUT FUNCTION

$o(t) \leftarrow p(t)$

FIGURE 4.14: ?-No-info Units
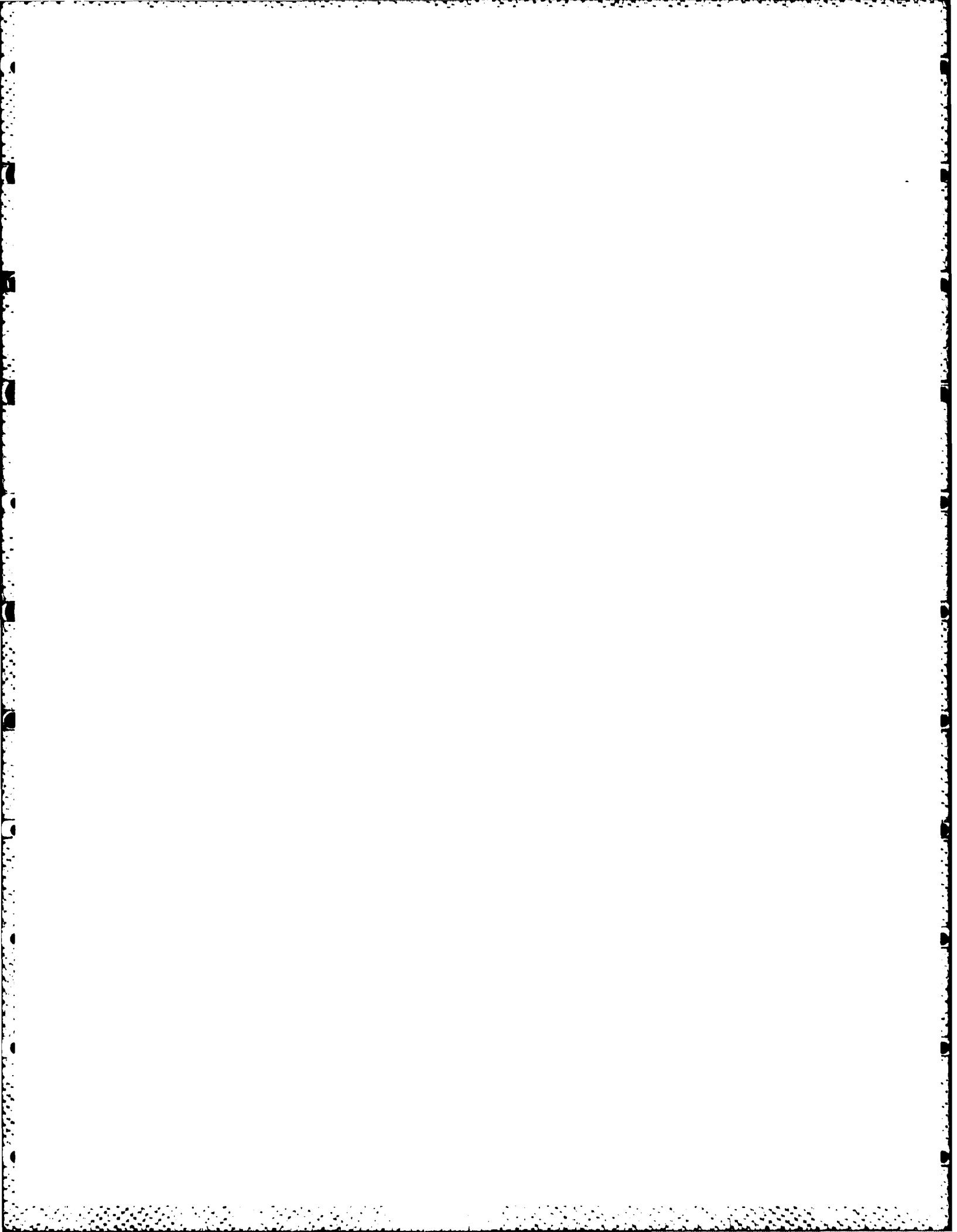
5. Open questions and future directions

This section briefly discusses several issues that have not been dealt with in the current paper. These include technical questions on evidence theory, convergence and learning. Furthermore, we propose to extend the current work in several directions and some of the possibilities are outlined below.

5.1 Evidence, Energy and Convergence

Throughout the earlier sections of the paper we have referred informally to our treatment of representation and inference as "evidential." The determining characteristics of an evidential treatment are quantitative inferences and the ability to deal effectively with incomplete information. In this section, we discuss some basic issues of evidential reasoning and how they articulate with our current implementation. Evidence theory is a subject of increasing importance in artificial intelligence and has many unresolved issues. From a certain point of view, a large part of AI and related fields depend crucially on a coherent method of combining evidence. This is particularly clear in Expert Systems efforts where combining beliefs is often the basic operation. But we can also view all perception problems as involving combinations of evidence for the presence of an edge, a word sense, an object, and so forth. Since there is no generally acceptable theory of evidential reasoning, the first question must concern how existing systems function at all.

Over-simplifying, one can say that existing evidential systems are either very small or do not rely heavily on their rules of combination. One can, in principle, eliminate general evidential reasoning by having a separate rule for each combination. Many expert systems are constructed along these lines and there are some attempts [Doyle 83] to establish that it should be done this way. The difficulties are the combinatorial explosion in the number of combination rules and the fact that adding a new statement involves deciding its interactions with all the existing ones. And no one has suggested using such an approach to low level perceptual tasks. Another reason why programs can work in the absence of a coherent evidence theory is that many decisions are sufficiently clear-cut to be insensitive to the detailed rules of combination. One can build vision systems that decide the presence of objects from evidence for features with almost any monotonic rule of combination. The other aspects of perception problems have been so difficult as to suppress the issue of evidential inference. But the issue is always there, and becomes especially important in connectionist treatments like the current paper. One can usually view each unit in a connectionist network as combining evidence (its input and state) and producing an output which can be treated as the unit's confidence in the validity of some proposition. A principled theory of unit behavior from an evidential reasoning standpoint appears to be necessary for the success of connectionist modelling.

To elucidate the notion of a formal evidence theory, we will focus on a particular rule called Dempster's Rule as expounded by G. Shafer in his book, *Theory of Evidence* [Shafer 76] and discussed in [Lowrance 82]. We will illustrate the rule with the Quaker-Republican example used earlier. We assume that believing someone is a Quaker provides subjective evidence that he is a pacifist and similarly for Republicanism. As mentioned above, the theory explicitly provides a way of expressing ignorance. Suppose we believe that 70-80% of Quakers are pacifists and

20-30% of them are not. The Dempster formulation would have us set the evidence weight for pacifist given Quaker, Bel (P|Q) to be .7, Bel (¬P|Q) = .2 and to assign the uncertainty formally to θ, the set of all possibilities under consideration. The formal variable θ corresponds to the residual uncertainty explicitly represented in e.g. the color of apples in Figure 3.12. Suppose our belief about the effect of Republicanism were Bel (P|R) = .2, Bel (¬P|R) = .6, Bel (θ|R) = .2. The Dempster-Shafer theory suggests a rule for combining these two sets of beliefs, given that they can be treated as independent evidence. [Shafer 76] presented the formal basis for the rule and discussed many of its properties: we will not attempt to summarize that book here. The intuitive content of the rule is best discussed in connection with Figure 5.1.

The outer parts of the table in Figure 5.1a simply encode the belief information given in the previous paragraph. Each box in the table is assigned a number that corresponds to the joint weight of its row and column and is something like a joint probability of two independent events. The crossed-out boxes capture the fact that a person is either a pacifist or not and so P and ¬P can't co-occur. The way to view the entries under θ is that ignorance is compatible with any event. Raw beliefs are formed by adding all the table entries for each event of interest. Since some joint events are logically precluded, the final belief structure is computed by normalizing with (dividing by) the sum of raw beliefs. The resulting answers are plausible, but do not make a compelling case for this particular rule of combination.

In fact, there are some unresolved issues with Dempster's rule (particularly on exactly when it applies), but it has a number of advantages for our purposes. An obvious advantage is that the set of beliefs for any question always adds to one, so that more evidence can be added in a uniform way. Suppose that we now learn that Dick is a member of the Marine reserves and believe that this suggests non-pacifism (.8) with θ = .1. Then the results of our previous calculation can be combined with this new evidence, incrementally (Figure 5.1b). The result would be a belief in Dick's non-pacifism of .77, of his pacifism equal .22, and uncertainty of .01. Similar calculations can be made using the *strength* of Dick's Quaker and Republican beliefs if these are known. This collection of features makes the Dempster-Shafer evidence theory a qualitative improvement over previous suggestions in AI for handling conflicting evidence [Quinlan 83] and suggest that the theory should be considered seriously by expert-system designers.

There are a number of other proposed rules for combining evidence and, in each case including Dempster's rule, there are clear situations where the rule is counter-intuitive. In fact, there is a considerable literature [Kyburg 74] on this subject, primarily by philosophers. Our current belief (sic) is that there are a modest number of distinct evidential situations for which different rules of combination are appropriate, but we have no idea whether they can all be treated as variations on a single principle such as maximum entropy. The point of current interest is that a formal evidence theory to specify the behavior of connectionist and other evidential systems is necessary and possible. Many of the detailed design decisions presented in Sections 3 and 4 are based on evidential considerations but we do not yet have a formal mapping from our implementation to any theory. One reason is that any connectionist implementation must also take into account the dynamics of network behavior.

Given that one has a way of specifying what a system should do, there remains the problem of showing that the goal is achieved. This dynamic correctness question for massively parallel systems has been receiving considerable attention and considerable progress is being made. A key insight of Hopfield [Hopfield 82] is that there is a close analogy between certain physical systems and connectionist networks whose desired behavior can be characterized by minimizing global energy functions. In this case, one can choose a unit rule that always reduces the global energy and thus insure convergence to at least a local minimum (of energy). The idea is easiest to see in a recognition situation [Geman & Geman 84]. The total energy could be made up of separate terms, each of which represents the compatibility of some hypothesis with the features found in the input. Ideally one would have all the relevant terms represented by separate units and arrange that each unit could compute its next value (of state or potential) in a way that reduced the total energy of the system. For example, a "line" unit that had inputs from many compatible "edge" units would assume a large (negative) value. Continued operation of the individual units would drive the system to a low energy state i.e. a solution. Of course, the situation is much more complicated than this (cf. [Hinton & Sejnowski 83]) and a number of serious technical problems remain to be solved. But the general idea of local operations that act to minimize a criterion is a fine way to look at the dynamics of connectionist networks.

One major difficulty with local relaxation schemes like that described above is that the system state can drift into a local minimum that is not an adequate solution to the original problem. The issue of a network converging to an incorrect solution is central in any formulation -- the energy function provides a convenient metaphor. A familiar example from perception is an image that is coarsely sampled, seen too close up. Your visual system detects edges that can't be interpreted. Moving back or squinting allows the system to reach a better overall solution by making it less confident of the spurious edges. The problem of false minima is inherent in connectionist networks because activation must spread through several levels in any non-trivial computation. There is no way for an irreversible local decision to be correct in all cases. There are a variety of ways to avoid local minima in recognition problems. One proposal is to use stochastic methods so that there is a high probablility of reaching a good solution [Hinton & Sejnowski 83]. Another idea is to design the energy function so that global organization has a large enough role to preclude stable local minima. Each of these has advantages and disadvantages. A third method, used in this paper, is to have the local computations move slowly in the direction perceived to be correct at each instant. The idea here is that distant computations have time to make their impact before a local configuration reaches a stable state. One pitfall in any dynamic system is oscillation and our design also avoids this problem.

In the current framework the controlled nature of spreading activation in the Memory Network and the restricted (only local) use of negative activation ensures that the nodes in the Memory Network will reach a stable level of activation. A look at the three kinds of queries supported by the network and the ensuing activity in the Memory Network should help in making this point. The use of enable signals on Binder and Relay nodes prevents spreading of activation that is irrelevant in the context of the query being processed. For each query, the activation flows along designated structural links only and converges to a few concept nodes (Values in case

of Class I queries and Owners in case of Class II queries). Each node accumulates the incident activation and the potential of these nodes increases in time to a steady state value. The only exception is the case where negative evidence inhibits a owner during the processing of a Class II query. However, even this does not result in an oscillatory behavior because the negative evidence impinging on a node is from strictly local nodes (its own Binders) and its contribution takes effect within a constant number of steps after the owner starts receiving positive activation.

Although there is no oscillatory behavior in the Memory Network, the problem of convergence and oscillations must also be considered in the context of the Answer Network. The source of oscillations in the Answer Network is conflicting evidence and incomplete information in the Memory Network. If two or more Response nodes in the Answer Network receive nearly equal activation from the Memory Network, there is a possibility of both the nodes rising to a high potential. The design of the Answer Network (as described in Section 4.4.3) is such that when this occurs, the growth of potential of all Response nodes is slowed down in the hope that eventually one of the answers will be able to dominate and win. This assumes that activation spreading along structural links may eventually activate parts of the Memory Network that may provide evidence to resolve the conflict. In case this does not happen within a reasonable interval the [?-conflict] node in the Answer Network begins to acquire a high potential and eventually wins. This mimics a time-out phenomena. A similar strategy is adopted to decide in the favor of [?-no-info] answer if none of the possible answers is getting any activation for a sufficiently long time. The duration of time-out is a design parameter and may be varied to meet specific design goals.

The current situation is that we are pursuing in parallel theories of evidence and convergence and the design of semantic neural networks. The current solutions to evidence and convergence problems have been robust over other changes in the system. The [?-conflict] node in answer networks deals with the dynamics issue of no clear answer. The [?-no-info] deals with the evidential issue of inadequate information. While there is no substitute for principled theories of behavior and dynamics for connectionist networks, we have been able to make progress in representation and inference questions within the current semi-formal framework.

5.2 Extracting answers from the Memory Network

In Section 3 we restricted the kinds of queries handled by the system to be multiple choice questions. We will now re-examine some important issues underlying this restriction.

Queries are posed to the Memory Network (network) by activating certain nodes and enabling specfic links in the network. Once the state of the network is thus initialized, the inference process proceeds independently in the network according to the built-in rules of propagation and evidence accumulation. The ensuing state of the network (primarily the levels of activation of concept nodes), in a sense, *is the result of the inference* performed in response to the query. However, the set of active nodes in the network resulting from a query not only includes nodes that correspond to the answer being sought, but also includes other nodes that take part in the inference process. For instance, in a query involving property inheritance many nodes along

the Type hierarchy may get activated before the node that represents the answer. With respect to Figure 5.2, a query such as ?v (B *Pl*) (What is the value of the property *Pl* of B), results in the activation of nodes C and D besides the node V1. The activation of nodes C and D may be viewed as intermediate steps in the inference process whereby the initial query ?v (B *Pl*) is succesively mapped into ?v (C *Pl*) and ?v (D *Pl*).

The presence of other active nodes besides the ones representing the answer raises the fundamental question of how to extract the final answer from the resulting state of the network. In the framework described in this paper we have finessed the problem of answer extraction by employing Answer Networks in routines. Routines always pose queries with reference to an explicit frame of discernment (the set of possible answers). The frame of discernment is encoded in the form of an Answer Network and the final answer is determined by the Answer Network node that receives the strongest support from the Memory Network.

Assuming the availability of an explicit frame of discernment is consistent with the notion of routines. Routines represent pre-wired (compiled) networks dedicated to specific tasks and hence it may be assumed that the possible answers to queries originating from routines are known in advance and encoded as Answer Networks within the routines. However, it is easy to visualize situations in which one cannot assume advance knowledge of the frame of discernment (in the present context - the existence of a routine with an appropriate Answer Network). A query such as "What does John like most?" does not have an obvious frame of discernment. The answers could be as varied as "ice-cream" (a kind of food), "science fiction" (a kind of literature), "tennis" (a sport) or even something such as "a glorious sunset" or "freshly fallen snow". The problem is further confounded in situations where the answer does not correspond to a specific node in the Memory Network but must be expressed by interpreting the relations between a number of active nodes. An example of this could be an answer such as: "the tall man wearing the black tie..." which would involve many active nodes; a probable set being nodes that represent MAN, HAS-HEIGHT, TALL, TIE, HAS-COLOR, BLACK and IS-WEARING.

In its most general form, the problem of answer extraction is related to and seems at least as complex as the problem of natural language generation. We feel that the work of other researchers in this area [McDonald 83; Dell 80; Simmons & Slocum 72] will provide us with valuble insights. Though at present we have not directed much effort towards solving this problem, we propose a possible way of dealing with some restricted cases of answer extraction in the absence of an explicit frame of discernment.

The kind of queries dealt with in this paper often define an implicit frame of discernment (*fod*) that consists of all instances of some Type represented in the Memory Network. Consider queries whose answer amounts to selecting an instance of a given Type on the basis of some specified property values. Such queries define an implicit *fod* consisting of the instances of the Type. For example, in the query: "Name a red tropical fruit", all the instances of the Type FRUIT constitute the *fod*. Other queries that specify a Concept (Type or Token) and a property and seek the value of the property may also define an implicit *fod* that consists of the possible values of the specified property. Often, the set of possible values correspond to a

single Type in the Memory network. For instance, query such as "What is the color of an Apple", implicitly defines a *fod* that consists of all the instances of the Type COLOR. However, all queries do not define an obvious *fod*; the query "What does John like most?" that may be interpreted as (?v (JOHN *HAS-LIKING-FOR*)), is a case in point. The cases in which an existing Type in the Memory Network constitutes a *fod* may be handled by requiring the design of the Memory Network to be such that all instances belonging to a Type be linked together in a WTA fashion. This amounts to having an Answer Network corresponding to each Type in the Memory Network - with the instances of the Type constituting the possible "answers". The WTA associated with the appropriate Type may be selectively enabled by routines during the processing of appropriate queries.

It is possible to extend the above idea to handle queries that have a very diffuse *fod* if any. This is done by using routines to perform a hierarchically organized search. If no *fod* is suggested by a query then the *fod* associated with the most general concept in the Memory Network is taken as the initial *fod*. As the computation progresses, the *fod* is refined incrementally by moving down the Type hierarchy until an acceptable answer is found. We illustrate this with the help of a simple example.

Figure 5.3 shows a simple hierarchy of Concepts in some Memory Network. Nodes enclosed in dotted lines form WTA networks. The WTA networks (WTA for short), are named after the immediate Type whose instances make up the nodes in the WTA. Thus, the WTA consisting of TENNIS and BASEBALL is referred to as [SPORT WTA]. Now imagine that the query "What does John like most?" is posed to the Memory Network. This will be done by activating JOHN and *HAS-LIKING-FOR*. As no *fod* is suggested by the query the WTA corresponding to EVERYTHING will be selected as the initial *fod*. Hence, the routine will enable [EVERYTHING WTA] consisting of the choices PHYSICAL-THING and NON-PHYSICAL-THING.

As a result of the routines activating JOHN and HAS-LIKING-FOR, the nodes ICE-CREAM, GLORIOUS-SUNSET, SCIENCE-FICTION and TENNIS will receive varying degrees of activation in proportion to the strengths of John's likings for each of them. These nodes will transmit activation up the Type hierarchy and eventually the nodes PHYSICAL-THING and NON-PHYSICAL-THING will also be activated. Without loss of generality, let us assume that NON-PHYSICAL-THING receives greater activation and wins the competition. Consequently, the *fod* now becomes [NON-PHYSICAL-THING WTA] with the choices being VISUAL-EXPERIENCE, LITERARY-KIND and SPORT. Continuing in this manner it is easy to see how in subsequent steps the *fod* may converge to [LITERARY-KIND WTA] with the choices being SCIENCE-FICTION and SHAKESPEAREAN-TRAGEDIES. If John likes SCIENCE-FICTION more than SHAKESPEAREAN-TRAGEDIES then it is easy to see how SCIENCE-FICTION will be chosen as the answer. Although we have not specified a termination criteria, one may imagine rules that terminate the process when the *fod* consists of nodes at the subordinate level [Rosch 75], or when one of the choices is above some absolute threshold.

Under the above proposal the routines essentially perform a breadth first search in parallel and the number of steps taken by even the most general query are proportional to the depth of the conceptual hierarchy. Admittedly the proposal needs

to be refined and we hope to do this in the near future.

5.3 Extending the representational framework

Our approach to the problem of semantic information requires us to treat the traditionally distinct issues of knowledge representation, inference and computational framework simultaneouly. In order to keep the over all complexity within manageable bounds while being honest to the approach, our strategy has been to consider only a restricted class of representational issues. This has allowed us to devote requisite attention to issues related to inference and the development of a connectionist system that embodies our solutions. In terms of purely representational issues we have thus far focused primarily on developing a framework that is best suited for representing simple concepts and natural kind terms. There are several important issues that we have not addressed as yet. These include representation of complex information such as description of actions, events, complex shapes, definition of composite relations, finer structure of properties and constraints between property values (structural descriptions [Brachman 79]). An open question is the division of knowledge between the Memory Network and the routines. Eventually, some of the information referred to above may be represented in the form of routines rather than in the Memory Network. Needless to say, many problems remain to be solved, but on the basis of our experience so far we are hopeful that it will be possible to extend the framework to solve most of the open questions. In this section we present a few assorted examples to indicate the kinds of issues being pursued.

Figure 5.4 shows the representation of the predicate LOVES. In the context of predicates it is easy to see the similarity in the notion of properties as used in our formulation and case roles that denote relations between predicates and noun phrases [Bruce 75; Fillmore 68]. The simplified representation in Figure 5.4 suggests that a PREDICATE has two case roles namely, *HAS-AGENT* and *HAS-PATIENT*. For the more specific predicate LOVES these cases roles get mapped into *HAS-LOVE-AGENT* and *HAS-LOVE-PATIENT* which in turn are filled by JOHN MARY in the representation of "John loves Mary".

In the example discussed above we used specific nodes such as *HAS-LOVE-AGENT* and *HAS-LOVE-PATIENT* as well as more general nodes such as *HAS-AGENT* and *HAS-PATIENT*. This corresponds to the use of *exploded cases*, a notion that has been found to be extremely useful in work on connectionist modeling of natural language processing [Cottrell & Small 83]. In Section 2 we over-simplified and used values such as red and green for the colors of apples and pears. However, we expect to represent these values by concepts that are much more fine grained. In developing representations it is important to bear in mind 'at the normal usage of language often belies the complexity of the information being communicated. In some cases detailed information may not be articulated as it is not relevant to the situation. However, oftentimes, a speaker does not make certain distinctions because he relies upon the hearer to make these by using his world knowledge. For instance, while refering to the color of an apple and that of a brick as "red" one seldom means that they are the one and the same color. One assumes that the hearer is aware of the difference between the two colors and hence will be able to interpret the two usages of "red" appropriately. In view of the above we intend to use color values such as

APPLE-RED, ROSE-RED and BRICK-RED. It is important to make these distinctions in a knowledge representation scheme in spite of the surface uniformity of language. Traditional knowledge representation systems do not have to represent these distinctions explicitly as they can shift this burden to the interpreter; the interpreter may be programmed to treat differently the value "red" when it is associated with distinct objects. The absence of an interpreter in our formulation, however, makes it necessary to explicitly represent concepts at a finer grain. We envisage the relationship between concepts such as APPLE-RED and RED to be the same as that between RED and COLOR. The properties associated with color — *HUE, BRIGHTNESS* and *SATURATION* — may be used to make classifications like RED and GREEN and also to make finer distinctions like BRICK-RED and APPLE-RED.

The role of exploded concepts acquires added importance in the representation of semantic information about events and actions. Finer case roles like *HAS-LOVE-AGENT, HAS-BUY-AGENT* and *HAS-PROPEL-AGENT* are needed to represent detailed information about and differences in predicates such as LOVE, BUY and PROPEL. Furthermore, distinct case roles make it possible to represent possible constraints on values of case roles. The hierarchical organization of concepts of varying granularity gives the ability to perform general inferences about COLOR and *HAS-AGENT* as well as specific inferences about APPLE-RED and *HAS-LOVE-AGENT*.

We also wish to pursue the representation of other ontological categories such as events, sets and situations. Different ontological categories have different sorts of attributes associated with them. For example, the representation of events could be based on properties such as *TIME-OF-OCCURRENCE, LOCATION-OF-OCCURRENCE, CAUSE-OF-OCCURRENCE* and *DESCRIPTION-OF-OCCURRENCE*. In modelling actions and events we hope to take advantage of the work by other researchers in AI and linguistics [Jackendoff 83; Bruce 75; Schank 73]. Figure 5.5 shows a simple example encoding the event described by "Jim made John hit Tom yesterday". As before, the figure is meant to convey a general idea of how we intend to approach these problems.

Sets and situations may be also be represented in a manner similar to that of other concepts. By sets we mean a finite and unordered collection of entities where the members of the set are explicitly enumerated. This corresponds to a naive notion of sets and is not equivalent to that of formal set theory. Like all other conceptual entities, sets are also represented as collections of <attribute, value> pairs. This collection includes a pair for each member of the set where the property in the pair is *HAS-MEMBER* and the value is one of the member concepts of the set. The collection of <attribute, value> pairs defining a set may also include structural links such as *is-a-subset-of* and *is-a-superset-of*. Inferences on sets will be controlled by specific routines that will compute set-membership, union and intersection.

A situation is a special kind of set consisting of entities, a set of relations on these entities and an associated location and time. Examples of situations are: "Harvard Square on a Friday night" or "an auction at Sotheby's". [Feldman 82b] describes how knowledge encoded as situations may be used during visual recognition. Situations may be represented in our formulation by extending the properties associated with sets to include the attributes *HAS-LOCATION* and *HAS-TIME-OF-OCCURRENCE*, and restricting the members of the set to be relations and entities

ocurring in these relations. The interactions between routines and situations remains to be worked out.

5.4 Learning

One standard criticism of connectionist models is that there is no plausible mechanism for the acquisition of knowledge. Although the problem of learning has not been solved for systems with a central interpreter and data structures, but there is clearly enough mechanism in this formulation to support learning. For connectionist modelers the problem is made more difficult by the biological constraint that new connections cannot be nearly rapid or extensive enough to account for everyday learning. The only mechanism available appears to be the change in the effectiveness (weight) of existing connections (synapses). Fortunately, there does seem to be adequate biological support for learning through weight change and there is a considerable literature on the mathematics of various possible alteration schemes. But all of this is focused on problems that are structurally much simpler than our routines and memory networks. The key technical issue is how a connectionist network could have a pre-existing structure rich enough to allow for learning the representations described in earlier sections of this report. Although we are far from solving this problem, we have a general idea of how learning may occur in the Memory Network. The learning of routines has not yet received serious attention.

The proposed mechanism for learning in the Memory Network is based on the notions of recruitment and chunking [Feldman 82a; Wickelgren 79] and we will discuss these in brief before outlining a plausible mechanism of concept formation. Broadly speaking, the idea of chunking may be described as follows: At a given time, the network consists of two classes of nodes:

> 1. *Committed* Nodes. These are nodes that have acquired a distinct "meaning" in the network. By this we mean that given any committed node, one can clearly identify sets of other committed nodes, whose activation will result in the former becoming activated. Committed nodes are connected to other committed nodes by "strong" links, and to a host of other *free* nodes, (see below), via "weak" links.

> 2. *Free* Nodes. These are nodes that have a multiplicity of weak links to other nodes, both free and committed. These form a kind of "primordial network" of uncommitted nodes within which the network of committed nodes is embedded.

Chunking involves strengthening the links between a cluster of committed nodes and a free node. Thereafter, the free node becomes committed and functions as the chunking node for the cluster i.e., the activation of nodes in the cluster results in the activation of the chunking node and conversely, the activation of the chunking node activates all the nodes in the cluster. The process by which a free node is transformed to a committed node is called *recruitment*. The mechanics of recruitment in connectionist networks is described in detail in [Feldman 82a]. The basic insight in the solution to the problem of learning through weight change is that certain classes of random connection graphs have a very high probability of containing the sub-network needed for learning a new concept.

The notion of chunking in its generic form only suggests a mechanism whereby nodes can be associated and is not sufficient for explaining how structured relationships arise. In the proposed solution we wish to exploit the non-trivial structure resulting from assuming that knowledge is organized in terms of properties and values thereof. We postulate that learning takes place within a network that is already organized to reflect this structure. For instance, in the context of vision, we specifically assume that concepts that correspond to primitive properties like color, shape, texture and motion are already present in the Memory Network of an agent together with concepts that represent some basic values of these properties. Simple forms of learning result in the formation of concepts that represent coherent collections of existing properties and values, while more complex forms of learning lead to generalization of concepts and the formation of complex properties that in turn lead to development of more comp... ...cepts.

We will consider a toy example of a Memor. Network interacting with a very simple visual system that is capable of detect... the colors blue and green and the primitive shapes round and oval. The initial organization of the Memory Network ... ... into account these characterstics of the visual system. Figure 5.6a is an ...simplified representation of the initial organization of the Memory Network. The ... ...ork has four pre-existing concepts namely, the property *HAS-COLOR* and its ...ues BLUE and GREEN and the property *HAS-SHAPE* and its values ROUND and OVAL. In other words, the nodes representing the properties and values are already connected to the visual system and may be activated by it under appropriate conditions. The nodes representing the four concepts are committed nodes embedded in a "primordial network" of free nodes that may be roughly partitioned into three diffused sub-networks X, Y and Z. Network X consists of nodes that are primarily connected to the nodes *HAS-COLOR*, BLUE and GREEN along with a host of free nodes in network Z. Nodes in network Y receive most of their connections from the nodes *HAS-SHAPE*, ROUND and OVAL and also from numerous free nodes in network Z. Finally, the nodes in network Z are connected to a large number of nodes throughout the Memory Network. The existence of networks X and Y indicates that the Memory Network is pre-wired to "know" that BLUE and GREEN are values of *HAS-COLOR* while ROUND and OVAL are the values of *HAS-SHAPE*.

...gure 5.6b depicts the result of learning an instance of a blue and round object. ... figure only shows the committed units and their interconnections. Learning an ...stance involves two stages of recruitment: the binder nodes B1 and R1 are ...cruited first, followed by the concept node BR1. When the visual system detects the color blue in the stimulus it activates the node *HAS-COLOR* and BLUE. The coincident activation results in the recruitment of a free node (B1) from the pool of free nodes in network X. The node R1 is recru... ... in an analogous manner from the pool of nodes in network Y. The simultaneous activity in B1 and R1 leads to the recruitment of the node (BR1) from network Z. Thereafter, the nodes B1 and R1 act as binder nodes and BR1 represents the newly acquired concept. B1 is activated by the coincident activity of *HAS-COLOR* and BLUE while R1 is activated by the coincident activity of *HAS-SHAPE* and ROUND. The activity of the concept node BR1 is strongly correlated with the activity of B1 and R1.

The working of the scheme depends on the assumptions we made about the pre-existing structure of the Memory Network. It was crucial to assume the existence of

property and value nodes with appropriate connections to the visual system. The organization of free nodes as networks X, Y and Z was equally important. Networks X and Y provided binder nodes in order to associate properties with their values, and the network Z provided a pool of nodes that could be recruited to "chunk" binder nodes in order to form concepts.

Figure 5.6c depicts the Memory Network with three instances (BR1, BR2 and BR3) of blue round objects and one instance (GO1) of a green oval object. In this situation a second kind of concept formation may occur and result in the formation of the concept "blue and round object" which is a generalization defined over BR1, BR2 and BR3. The resulting network is shown in Figure 5.6d. The new concept is represented by the node BR that owns the binders B and R that indicate its property values. These property values correspond to the shared property values of the instances.

The transformation from the network in Figure 5.6c to that in Figure 5.6d is best explained with the help of the simpler networks shown in Figure 5.7. The network shown in Figure 5.7b is the result of a similar transformation of the network in Figure 5.7a. The three instances A, B and C have the same value (V) for the property P and this forms the basis for the formation of the more general concept D. The transformation occurs in two stages.

I. A chunking node for b1, b2 and b3 is recruited from a pool of free nodes that serves the same function as network Z in the previous example, i.e. provides a potential concept node.

II. i) Over a longer period of time, the multiple paths between P and V via b1, b2 and b3 collapse into a single path via b, where b is one of the existing binder nodes b1, b2 or b3. The collapsing of links does not mean that the links disappear, but rather that the weights of links get reduced in such a way that all binder nodes besides b, gradually become free nodes (are released).

ii) The connection between b and D remain strong but the connections between other binder nodes and D become weak.

iii) The links x,y and z (in effect) now emanate from D rather than the binder nodes.

(All changes described in stage II happen during the same time interval).

The net effect of I and II is that the network shown in Figure 5.7a behaves like the network shown in Figure 5.7b. The scheme that we have just described characterizes learning as network transformations that minimize the complexity of the network (number of links and nodes) while maintaining the cause effect relationships between existing concept nodes. Thus, the nodes P, V, A, B and C have roughly the same effect on each other in the two networks shown in Figures 5.7a and 5.7b. The complexity of networks is substantially reduced by formation of more general concepts although this may not be evident from this simple example. In general, if the generalization takes place over p properties and c instances (the values

of p and c were 1 and 3 in the example of Figure 5.7 and 2 and 3 in the example of Figure 5.6d), then the savings in the number of links and nodes is of the order of p×c.

Referring back to Figure 5.6d. BR, a node in network Z, will be recruited as a chunking node of B1, B2, B3 as well as R1, R2 and R3. The release of binder nodes and the collapsing of links will occur separately for the two properties *HAS-COLOR* and *HAS-SHAPE*. Thus, B1, B2 and B3 will collapse into B while R1, R2 and R3 will collapse into R.

We have only provided a crude description of how recruitment of free nodes and release of committed nodes gives rise to representation of new instances and development of concepts that are generalizations of existing concepts. The latter kind of concept formation is accompanied by substantial reduction in the number of committed nodes and links. We hope to refine some of these ideas in the near future. A more detailed account will appear in [Shastri 84].

We expect that we will not require major changes in the *basic* design of our networks in order to support learning. The connectionist implementation described in this paper uses some complex unit types but it is possible to implement the same basic design in terms of simpler unit types that are more likely to fit into a learning scheme. The primary reason for not using the simpler unit types was to keep the simulations simple. The use of simpler unit types would result in larger networks because more than one simple unit would be required to perform a function currently performed by a single unit. Probably the best way to view this subsection on learning and the other parts of Section 5 is as plausibility arguments suggesting that there are no insuperable barriers to a complete connectionist theory of semantic memory. We would greatly appreciate comments on problems we have overlooked or on any other aspect of the report.

Figure 5.1

REP. / QUAKER	P = .2	¬P = .6	Θ = .2
P = .7	.14		.14
¬P = .2		.12	.04
Θ = .1	.02	.06	.02

raw beliefs: P = .3;   ¬P = .22;   Θ = .02

normalized beliefs: P = .55;   ¬P = .41;   Θ = .04

5.1a

MARINE / REP. Θ QUAKER	P = .1	¬P = .8	Θ = .1
P = .55	.055		.055
¬P = .41		.328	.041
Θ = .04	.004	.032	.004

raw beliefs: P = .11;   ¬P = .40;   Θ = .004

normalized beliefs: P = .22;   ¬P = .77;   Θ = .01

5.1b

Figure 5.1   Dempster applied to Dick
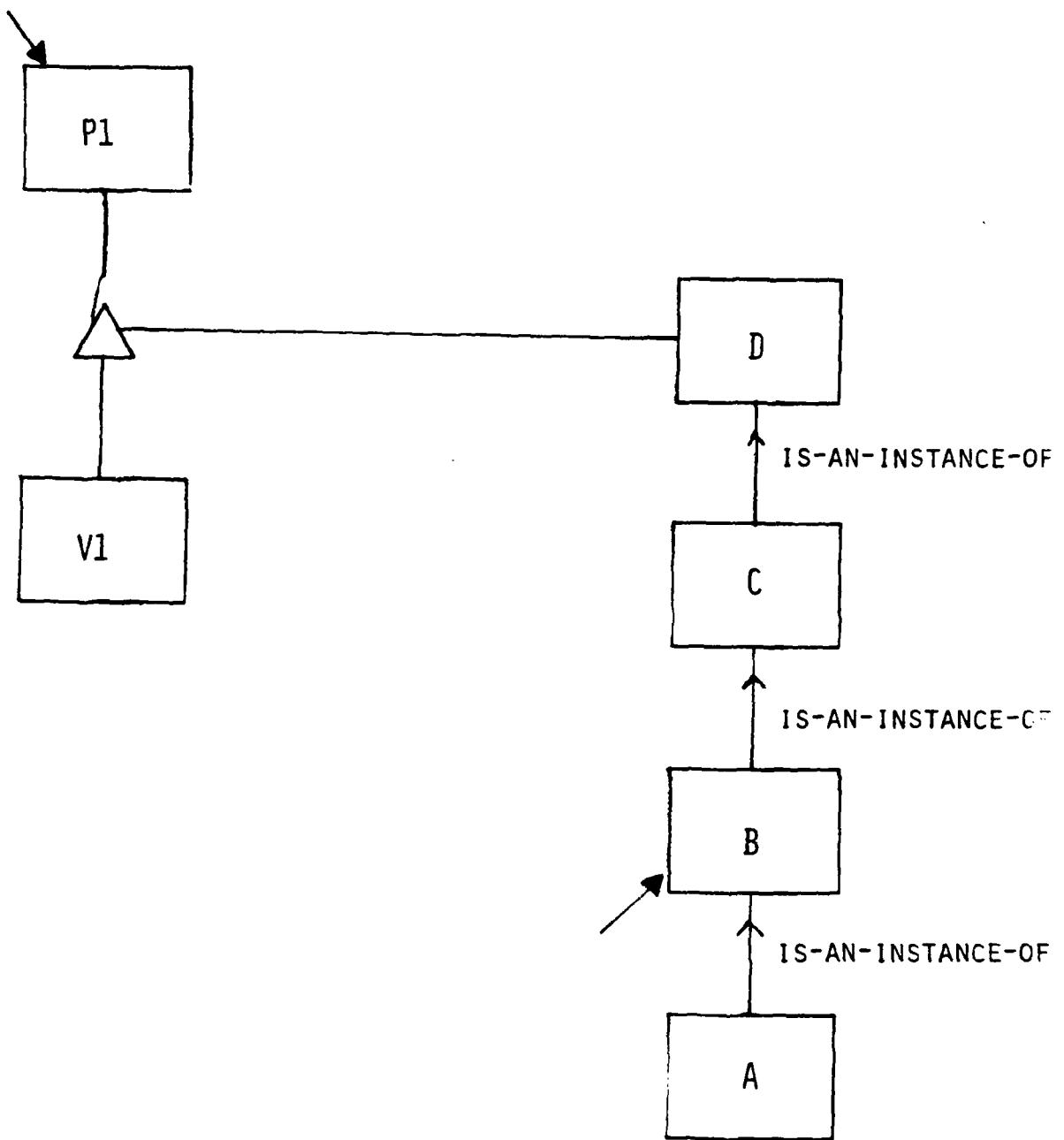
QUERY: ?v (B P1)   THE QUERY ACTIVATES THE IS-AN-INSTANCE-OF
LINKS AND THE NODES MARKED THUS ⟶ .

FIGURE 5.2
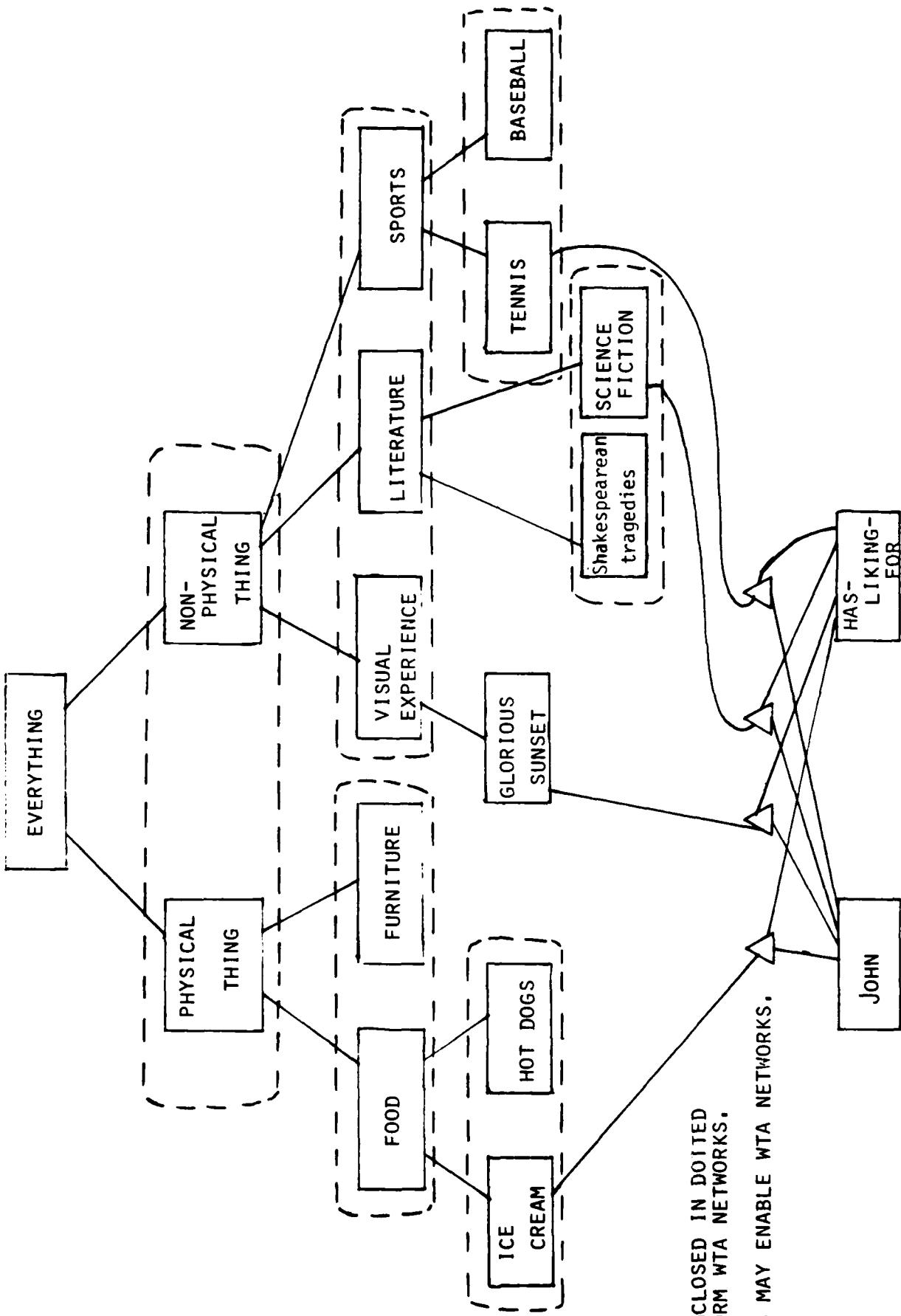
FIGURE 5.3

NODES ENCLOSED IN DOTTED
LINES FORM WTA NETWORKS.

ROUTINES MAY ENABLE WTA NETWORKS.
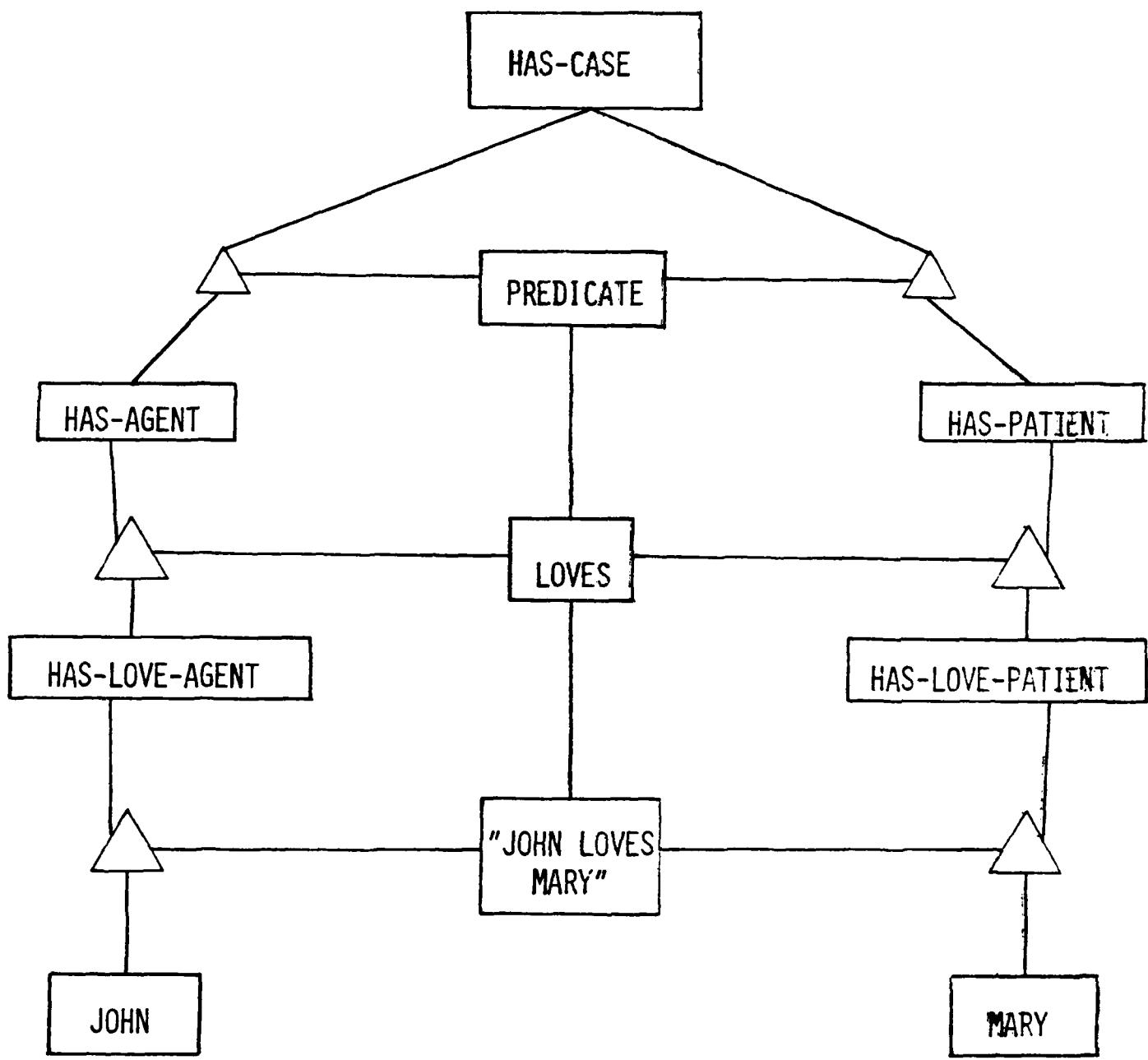
FIGURE 5.4   "JOHN LOVES MARY"
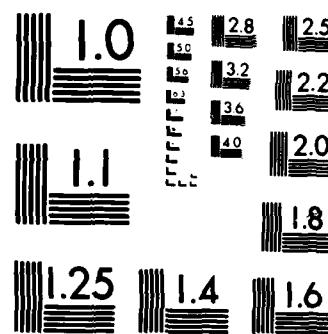
MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A
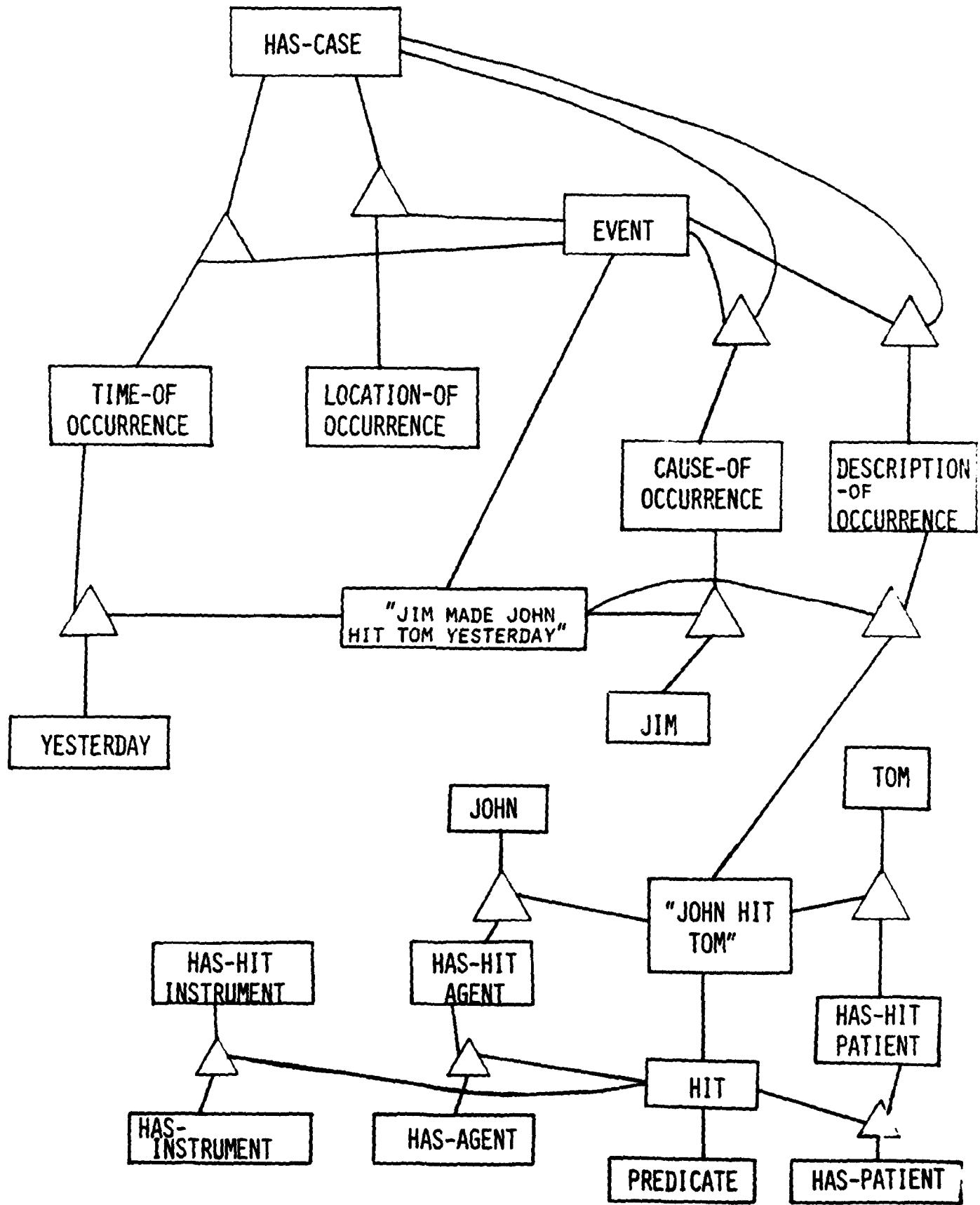
FIGURE 5.5 Jim made John hit Tom yesterday

PROPERTIES

HAS-SHAPE

HAS-COLOR

VALUES

OVAL

ROUND

GREEN

BLUE

--- WEAK LINKS
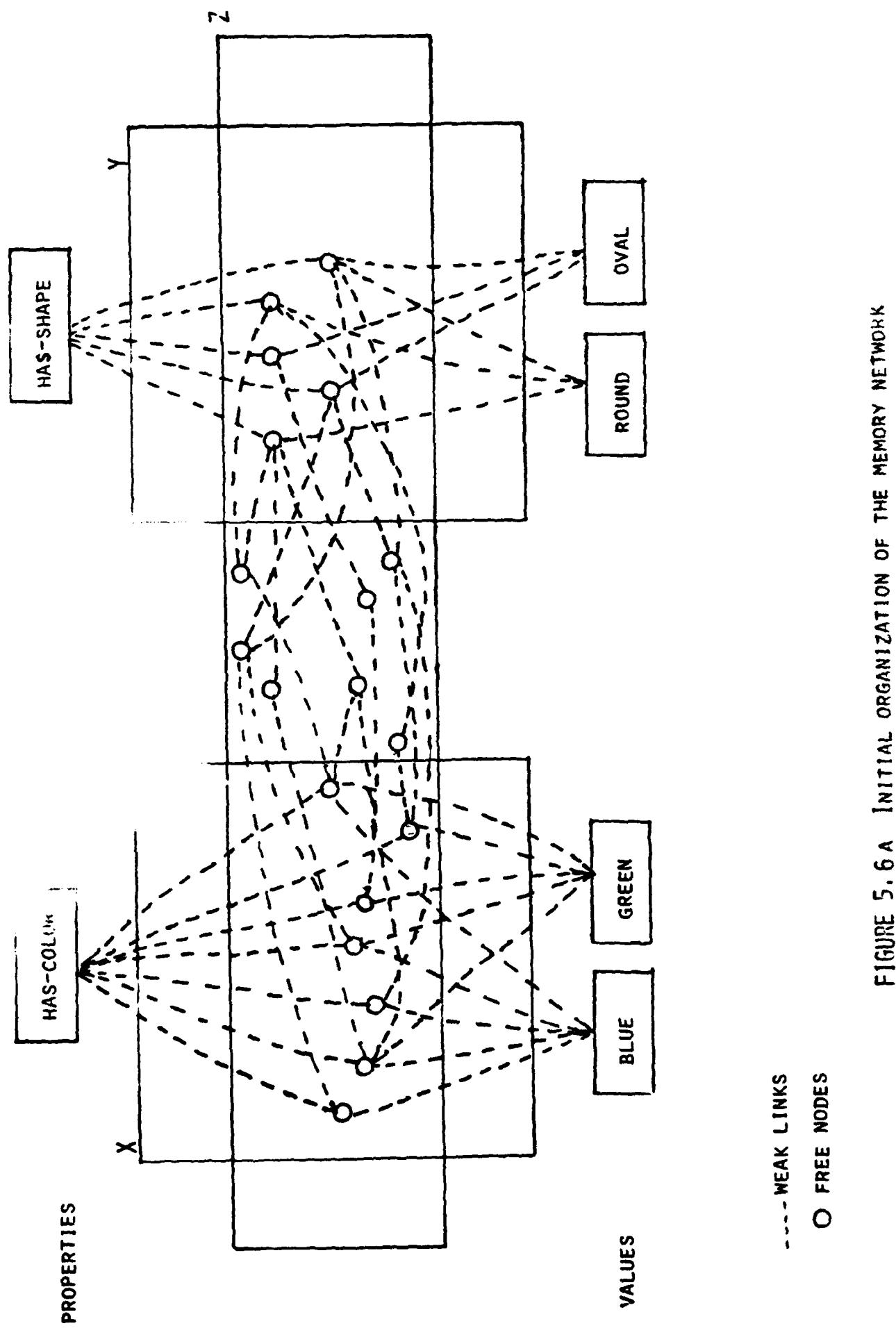
O FREE NODES

FIGURE 5.6A INITIAL ORGANIZATION OF THE MEMORY NETWORK

PROPERTIES

HAS-SHAPE

HAS-COLOR

BR1

R1

B1

VALUES

OVAL

ROUND

GREEN

BLUE

FREE NODES AND WEAK LINKS ARE NOT SHOWN.
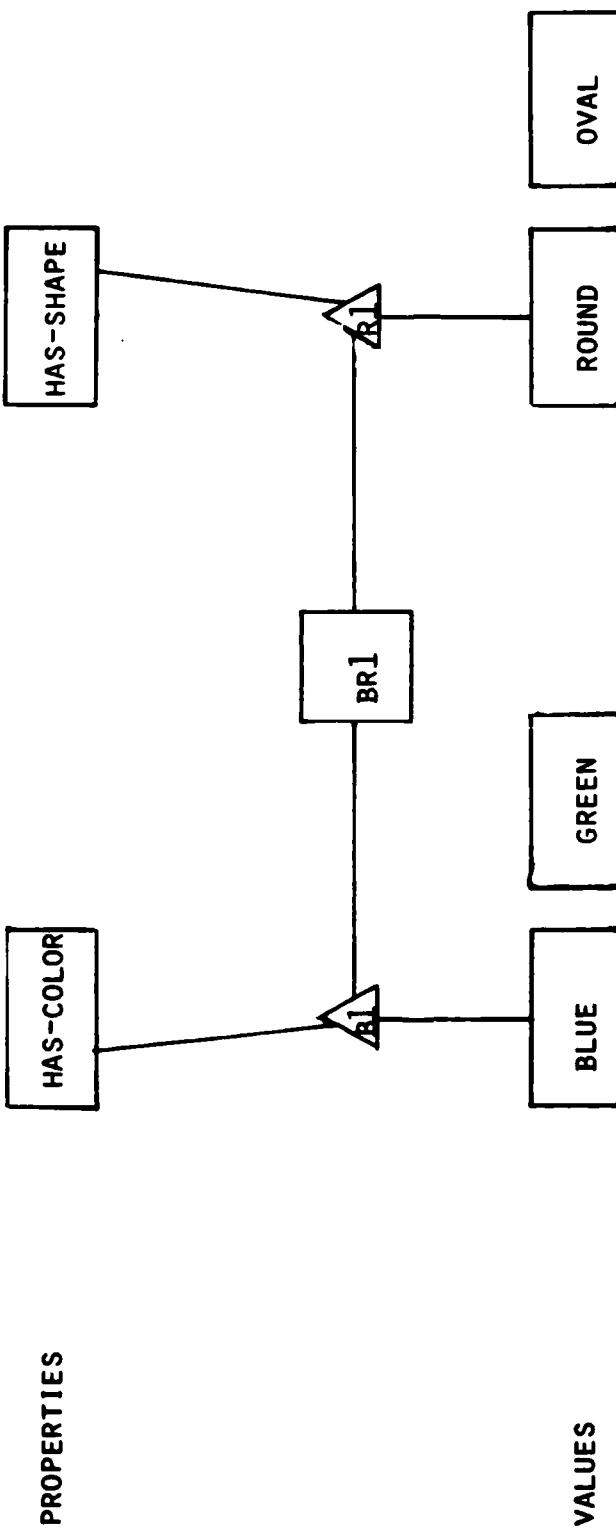
FIGURE 5.6 B A BLUE AND ROUND OBJECT REPRESENTED IN THE MEMORY NETWORK
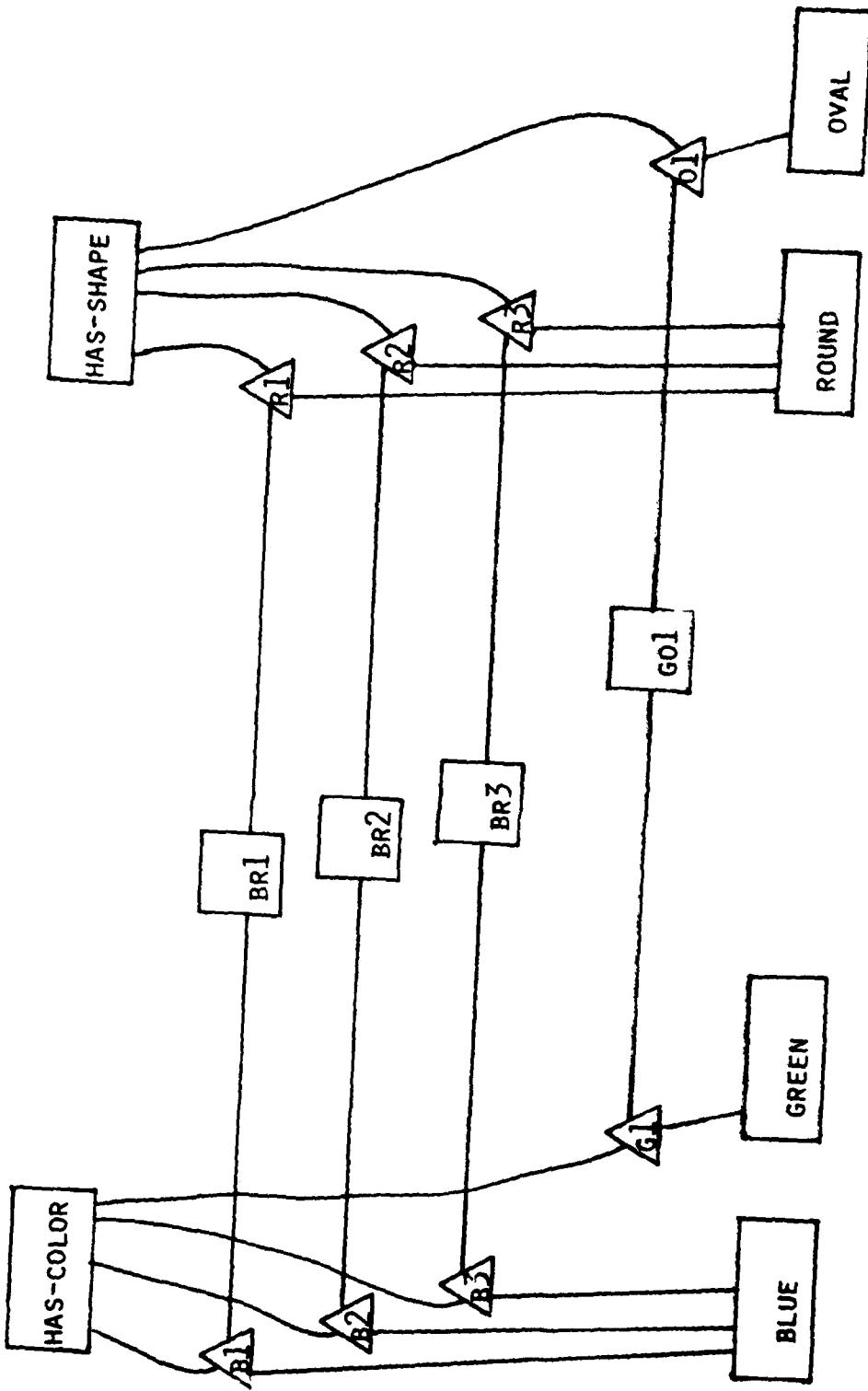
FIGURE 5.6.c

FREE NODES AND WEAK LINKS NOT SHOWN.
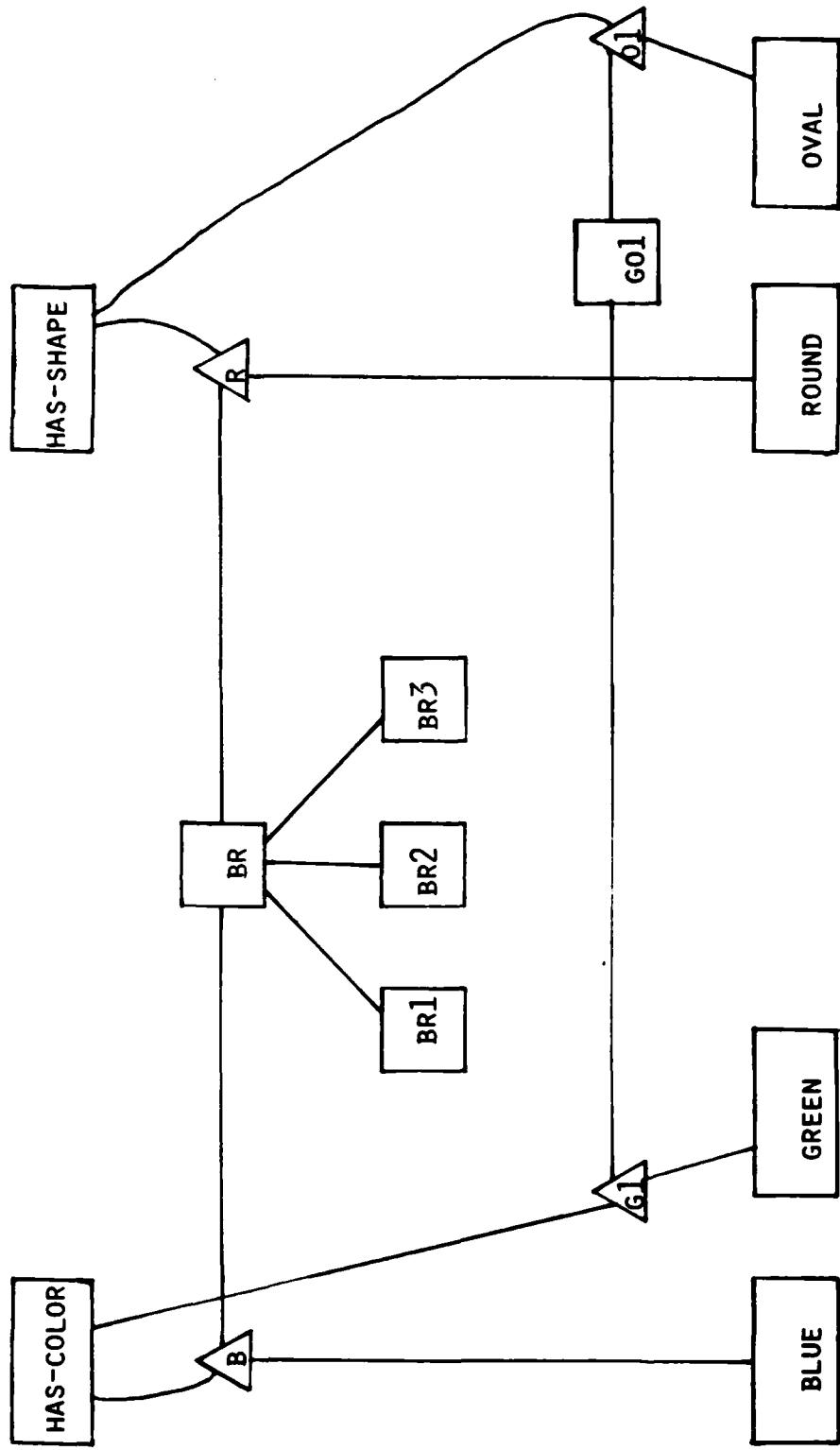
FIGURE 5.6 D

FREE NODES AND WEAK LINKS ARE NOT SHOWN
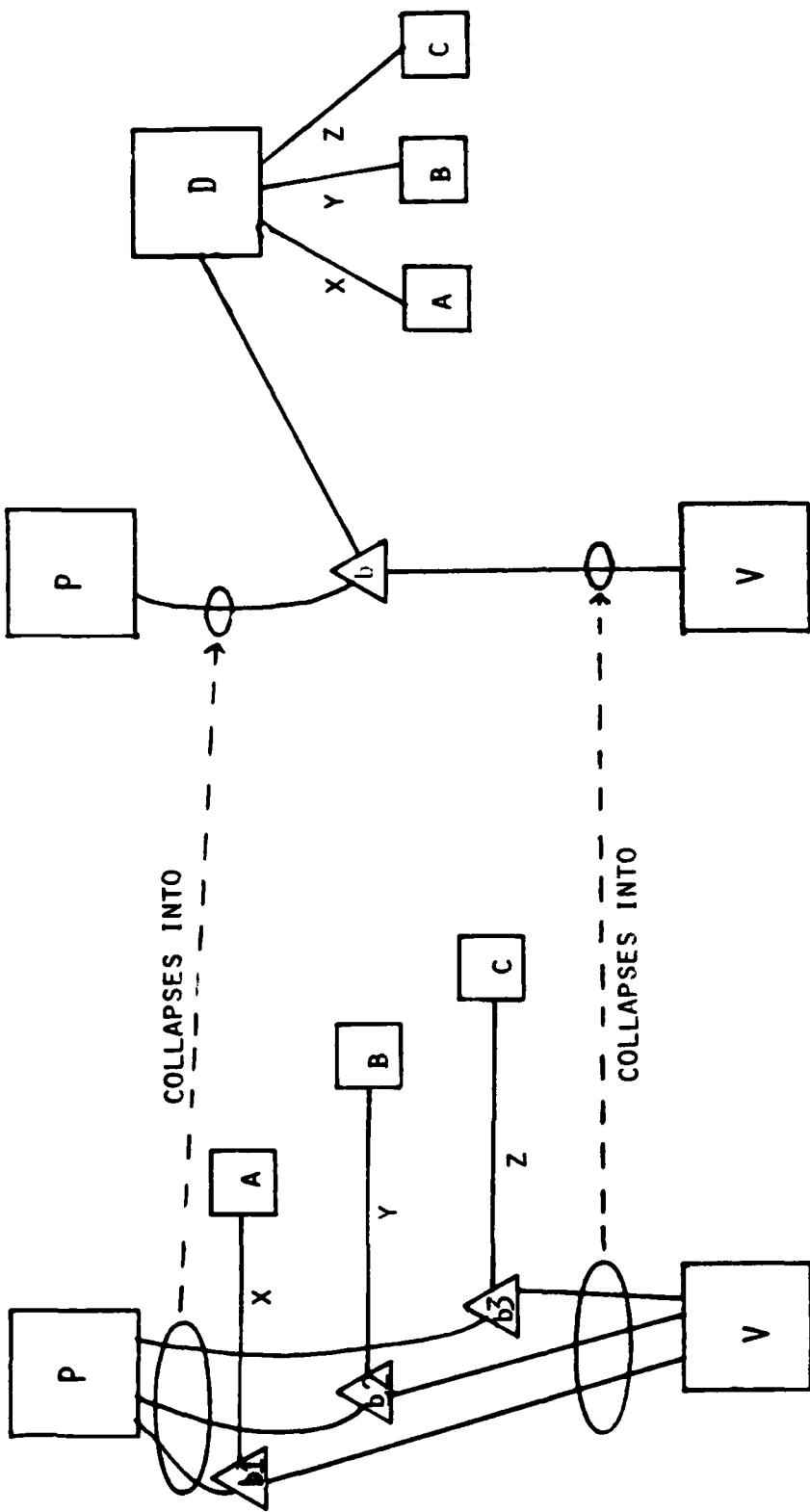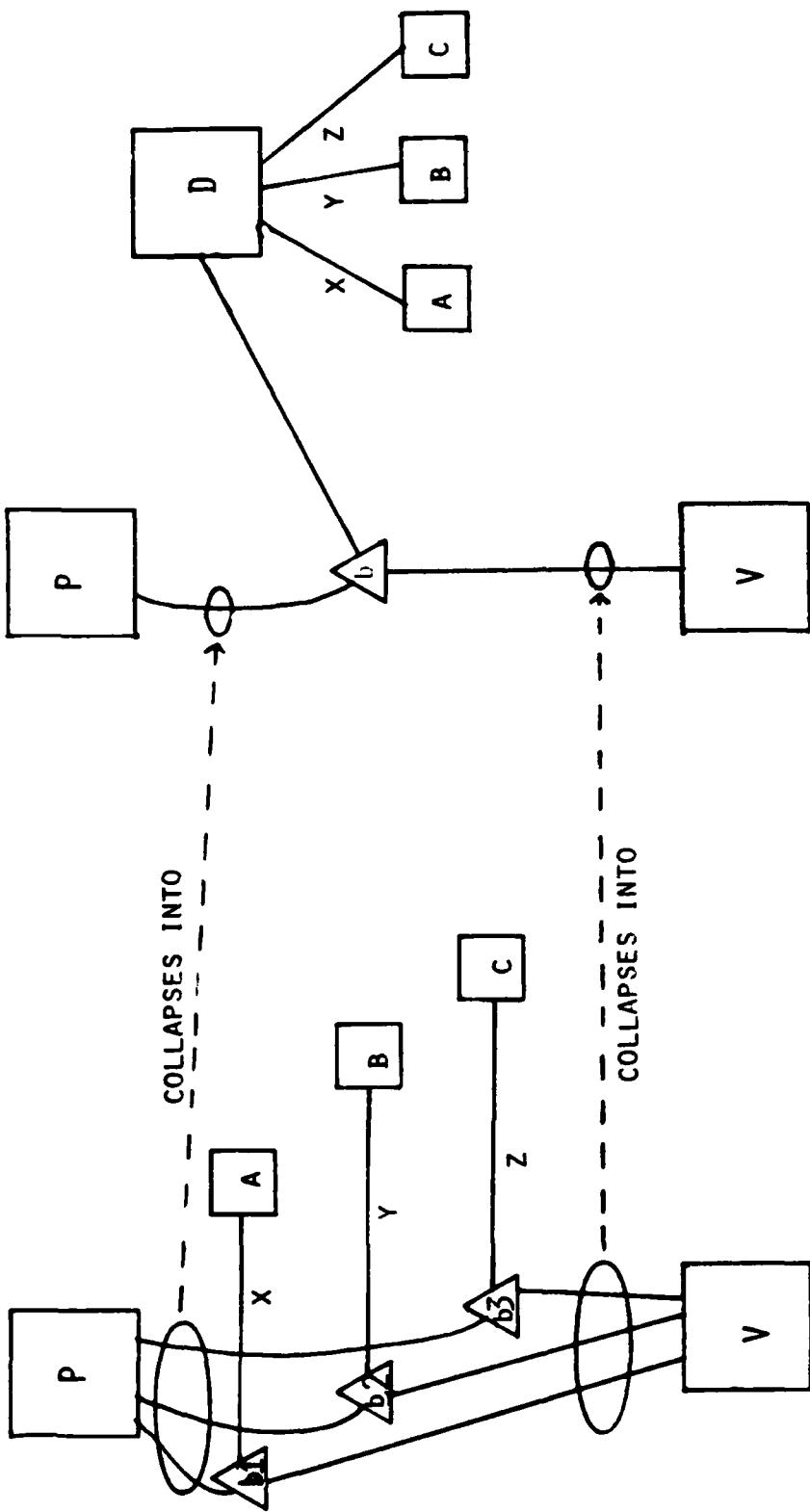
Figure 5.7a

Figure 5.7b

FIGURE 5.7 NETWORK IN FIGURE 5.7A COLLAPSES INTO NETWORK IN FIGURE 5.7B

BIBLIOGRAPHY

[Allen 83] Allen, James F. Maintaining Knowledge about Temporal Intervals. *Commun. ACM 26*, 1983, 832-843.

[Addanki 83] Addanki, Sanjay. Applications of Connectionist Modeling for Motor Control Systems. Ph.D. Dissertation. Dept. of Computer Science. University of Rochester, 1983.

[Anderson 83] Anderson, John R. *The Architecture of Cognition.* Harvard University Press. Cambridge Massachusetts. 1983.

[Ballard & Brown 82] Ballard, Dana H., and Christopher M. Brown, *Computer Vision.* Prentice Hall, Englewood Cliffs, New Jersey. 1982.

[Bobrow & Winograd 76] Bobrow, Daniel G., and Terry Winograd. An Overview of KRL: A Knowledge Representation Language. CSL-76-4. Xerox Palo Alto Research Centre.

[Brachman 82] Brachman, Ronald J. What "ISA" Is and Isn't. *Proceedings of the Fourth National Conference of the Canadian Society for the Computational Studies of Intelligence.* Saskatoon, Canada. 1982. pp 212-221.

[Brachman 77] Brachman, Ronald J. A Structural Paradigm for Representing Knowledge, Ph.D. Thesis. Division of Engineering and Applied Physics, Harvard University, 1977.

[Bruce 75] Bruce, B.C. Case Systems for Natural Language. *Artificial Intelligence,* 1975, 6, 327-360.

[Cottrell & Small 83] Cottrell, Garrison W, and Steven L. Small. A Connectionist Scheme for Modeling Word Sense Disambiguation. *Cognition and Brain Theory,* 6(1), 89-120.

[Dell 80] Dell, Gary S. Phonological and Lexical Encoding in Speech Production. Ph. D. dissertation, Dept. of Psychology, University of Toronoto, 1980.

[Dosher 83] Dosher, Barbara Anne. Effect of Sentence Size and Network Distance on Retrieval Speed. In *Journal of Experimental Psychology: Learning, Memory, and Cognition.* vol. 8, No. 3, 173-207. 1983.

[Doyle 83] Doyle, Jon, Some Theories of Reasoned Assumptions: An Essay in Rational Psychology. CS-83-125, Technical Report Carnegie-Mellon University, Pittsburgh PA.

[Fahlman 82] Fahlman, Scott E. Three flavors of Parallelism. *Proceedings of the 4th National Conference of the Canadian Society for Computer Studies of Intelligence,* 1982.

[Fahlman, Touretzky & van Roggen 81] Fahlman, Scott E., Touretzky David S., and Walter van Roggen. Cancellation in a Parallel Semantic Network. In *The proceedings of 7th International Joint Conference on Artificial Intelligence,* Vancouver. B.C. 1981.

[Fahlman 79] Fahlman, Scott E. *NETL: A System for Representing and Using Real-World Knowledge.* The MIT Press, 1979.

[Feldman 82a] Feldman, Jerome A. Dynamic Connections in Neural Networks, *Biological Cybernetics*, 46, 27-39, 1982.

[Feldman 82b] Feldman, Jerome A. Four Frames Suffice: A provisionary model of vision and space. TR99. Dept. of Computer Science, University of Rochester, 1982.

[Feldman & Ballard 82] Feldman, Jerome A., and Dana H. Ballard. Connectionist Models and Their Properties. *Cognitive Science*, 6, pp 205-254, 1982.

[Fillmore 68] Fillmore, C.J. The Case for Case. In *Universals in Linguistic Theory*, E.W. Bach and R.T. Harms (Eds.), pp 1-88. Holt, Rinehart & Winston, New York. 1968.

[Findler 79] Findler, Nicholas V. (Ed.) *Associative Networks: Representation and Use of Knowledge by Computers.* Academic Press. New York. 1979.

[Fox 82] Fox, Mark S. Reasoning with incomplete knowledge in a resource-limited environment: Integrating reasoning and knowledge acquisition. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C. pp 313-318.

[Geman & Geman 83] Geman, Stuart and Donald Geman. Stochastic Relaxations, Gibbs Distribution and the Baysian Restoration of Images. Unpublished manuscript September 1983.

[Hinton & Sejnowski 83] Hinton, G.E., and T. Sejnowski. Analyzing Cooperative Computation. *Proceedings of the Fifth Annual Conference of the Cognitive Science Society.* Rochester, New York. May 1983.

[Hopfield 82] Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences USA*, 1982, 79 pp 2554-2558.

[Jackendoff 83] Jackendoff, Ray. *Semantics and Cognition.* The MIT Press, Cambridge Massachusetts. 1983.

[Kyburg 74] Kyburg, Henry. *The Logical Foundations of Statistical Inference.* D. Reidel Publishing Company, Dordretchi-Holland, Boston USA. 1974.

[Lowerre & Reddy 79] Lowerre, B.T. and R. Reddy. The HARPY speech understanding system. In W.A. Lea (Ed.) *Trends in Speech Recognition.* Prentice Hall, Englewood California. 1979. Chapter-15.

[Lowrance 82] Lowrance, John D. Dependency-Graph Models of Evidential Support, Ph. D. dissertation, Department of Computer and Information Science, University of Massachusetts, Amherst, Mass. (September 1982).

[Marr & Nishihara 78] Marr, D., and H.K. Nishihara. Representation and Recognition of the Spatial Organization of Three-dimensional Shapes. *Proceedings* of the Royal Society of London B 200, 1978, 269-294.

[McClelland & Rumelhart 81] McClelland, J.L, and D.E. Rumelhart. An Interactive Activtion Model of Context Effects in Letter Perception. Part I, An account of basic findings. *Psych. Review*, 1981, 88, 375-407.

[McDonald 83] McDonald, David D. Natural Language Generation as a Computational Problem: An Introduction. In M. Brady & R.C. Colby (Eds.) *Computational Models of Discourse*. Chapter-4. The MIT Press, Cambridge Massachusetts, London England. 1983

[Minsky 75] Minsky, Marvin. A Framework for Representing Knowledge. In P.H. Winston *(Ed.), The Psychology of Vision*. McGraw Hill, 1975.

[Quillian 68] Quillian, Ross M. Semantic Memory. In M. Minsky (Ed.), *Semantic Information Processing*. MIT Press, 1968.

[Quinlan 83] Quinlan, J.R. Consistency and Plausible Reasoning. *Proceedings of the Eight International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany. 1983. pp 137-144.

[Reiter 80] Reiter, Raymond. A Logic for Default Reasoning. *Artificial Intelligence*, 13 pp 81-132.

[Roberts & Goldstein 77] Roberts. Bruce. and Ira Goldstein. The FRL Manual MIT AI memo 409, 1977.

[Rosch 75] Rosch, E. Cognitive Representations of Semantic Categories. *Journal of Experimental Psychology*: General 104, 192-233, 1975.

[Shafer 76] Shafer, G. *A Mathematical Theory of Evidence*. Princeton, N.J.: Princeton University Press, 1976.

[Schank 73] Schank, Roger C. Identification of Conceptualization Underlying Natural Language. In R.C. Schank and K. Colby (Eds.). *Computer Models of Thought and Language*. Freeman, San Francisco 1973.

[Schank 82] Schank, Roger C. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. New York Cambridge University Press. 1982.

[Shastri 84] Shastri, Lokendra. Knowledge Representation in a Parallel Evidential framework. Forthcoming Ph.D. Dissertation. Dept. of Computer Science. University of Rochester.

[Simmons & Slochum 72] Simmons, R.F, and J. Slochum. Generating English Discourse from Semantic Networks, *Comm. of the ACM 15*, 10 891-905. 1972.

[Small *et al.* 82] Small, S.L., Shastri L., Brucks M.L., Kaufman S.G., Cottrell G.W., and S. Addanki. ISCON: An Interactive Simulator for Connectionist Networks. TR 109. Dept. of Computer Science, University of Rochester, December 1982.

[Smith & Medin 81] Smith, Edward E., and Douglas L. Medin. *Categories and Concepts*. Harvard University Press, Cambridge Massuchusetts. 1981.

[Walker 78] Walker, D.E. (Ed.) *Understanding Spoken Language*. American Elsevier. New York. 1978.

[Wickelgren 79] Wickelgren, Wayne A. Chunking and Consolidation: A Theoretical Synthesis of Semantic Networks, Configuring in Conditioning, S-R Versus Cognitive Learning, Normal Forgetting, the Amnesic Syndrome,

and the Hippocampal Arousal System. In *Psychological Reviews*. Vol. 86, No. 1, 44-60., 1979.

END

FILMED

3-85

DTIC